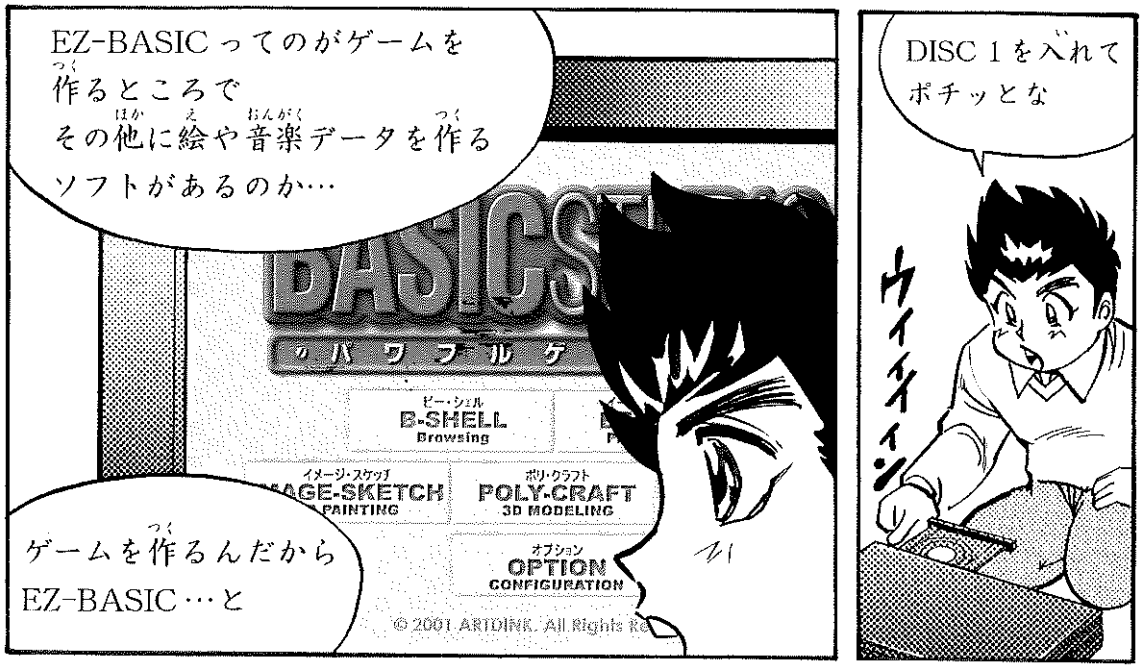
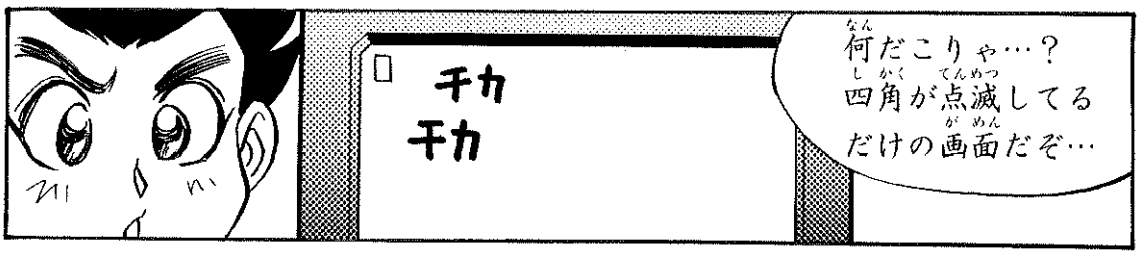
はじめよう！

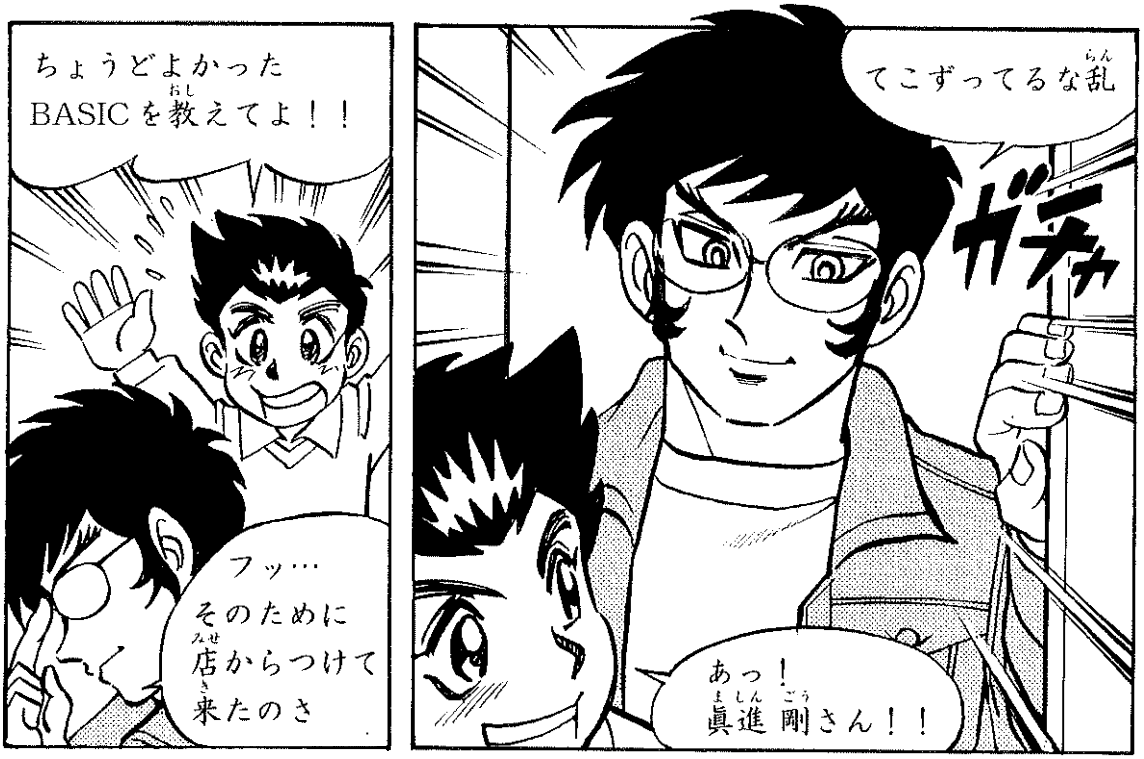
EZ-BASIC

第一章

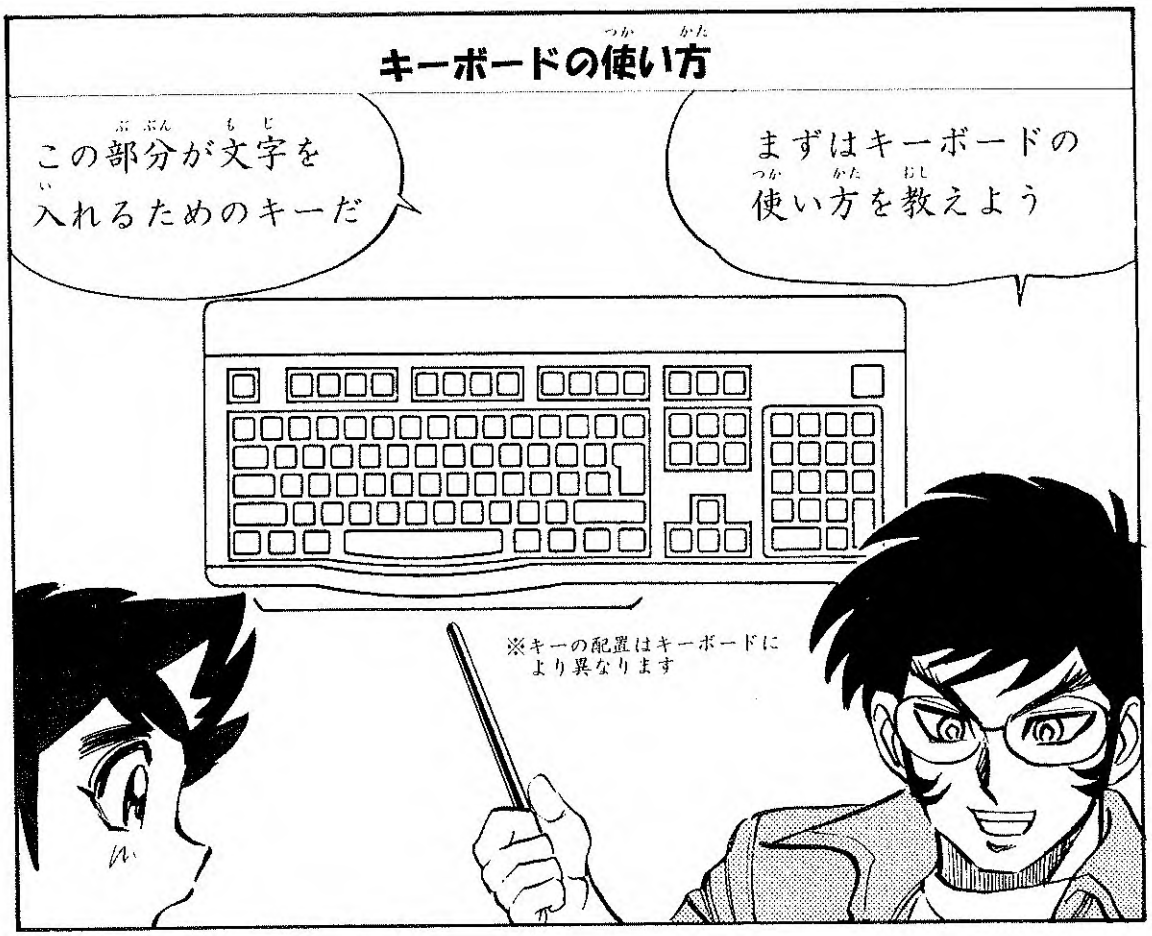
※アナログコントローラ(DUALSHOCK 2)でも同様な操作ができます。


EZ-BASICってのがゲームを作るところでその他に絵や音楽データを作るソフトがあるのか…
BASICS
のパワフルゲ
B-SHELL
Browsing
IMAGE-SKETCH
PAINTING
POLY-CRAFT
3D MODELING
OPTION
CONFIGURATION
© 2001 ARTDINK. All Rights Re
ゲームを作るんだからEZ-BASIC…と
DISC 1を入れてポチッとな
ウィィィン

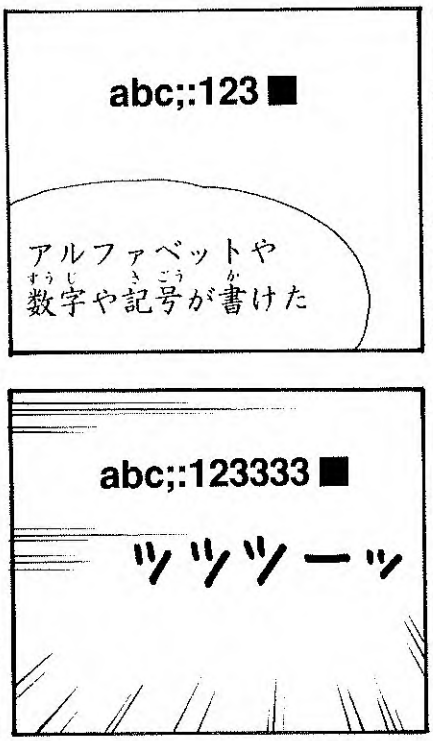

チカ
チカ
何だこりゃ…？四角が点滅してるだけの画面だぞ…

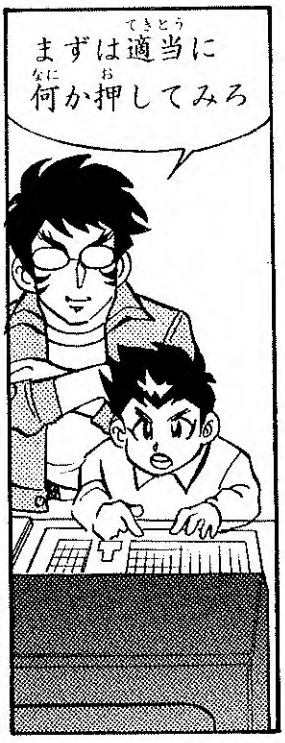

ちょうどよかったBASICを教えてよ！！
フッ…そのために店からつけて来たのさ
てこずってるな乱
ガチャ
あっ！真進剛さん！！

キーボードの使い方


この部分が文字を入れるためのキーだ
まずはキーボードの使い方を教えよう
※キーの配置はキーボードにより異なります

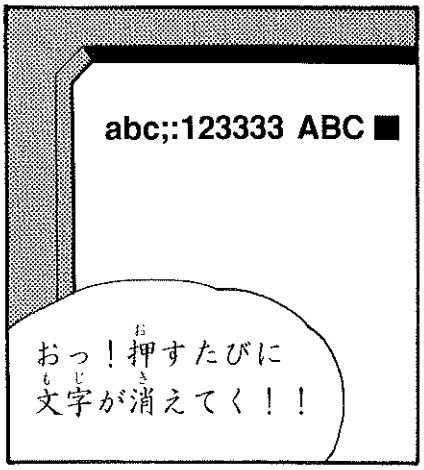
キーを押したままにすると連続で文字が書けるのか…
それがオートリピートという機能だ

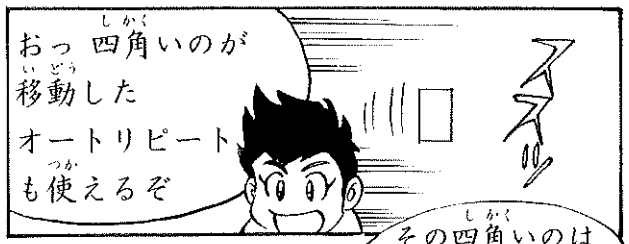

abc;:123■
アルファベットや数字や記号が書けた
abc;:123333■
ツツツーッ

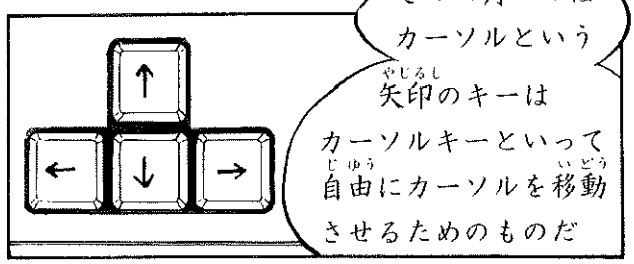

まずは適当に何か押してみろ

1つのキーに幾つかの記号が書かれてるやつもあるぞ？
あ
3あ
$ う
4う
%
5
真ん中には何も書かれていないキーがある
そいつはスペースキーだ空白をあけるときに使う
Shiftキーを押しながらさっきのキーを…
ぽち
ぽちっ
キーボードの横の方にあるShiftキーを押しながらやってみろ
おっ！アルファベットは大文字に変わって数字のキーとかは上側に書いてある記号になった！！
abc;:123333 ABC!"#$%
今度はBack Spaceってキーを押してみろ
Back Space
Enter
キーボード上隅のCaps Lockってランプが光った
それでShiftキーを押さなくても大文字が書けるぞ
Shiftを押しながらCaps Lockキーを押してみろ

そいつは一文字戻って
文字を消すキーだ
押したままにすれば
オートリピートで
どんどん文字が消えるぞ


abc;:123333 ABC ■
おっ！押すたびに
文字が消えてく！！

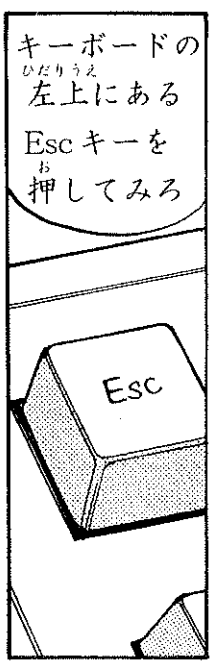

おっ 四角いのが
移動した
オートリピート
も使えるぞ

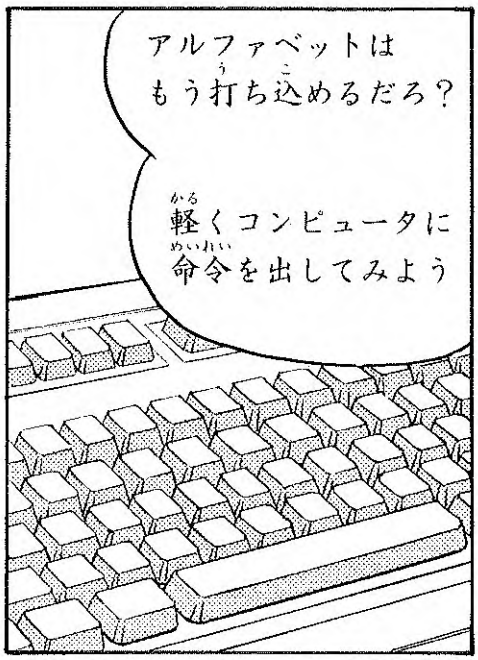

その四角いのは
カーソルという
矢印のキーは
カーソルキーといって
自由にカーソルを移動
させるためのものだ

次は上下左右の矢印
が書いてあるキーを
押してみろ
ほいっ
ポチッ

プログラムを作っていると
何かと文字で画面が見難くなる
からこれで文字を全部消して
見やすくすると良い
とりあえずキーボードに
ついてはこれだけ覚えて
いれば大丈夫だ
うし！！

■
おおっ！！
画面の文字が
みんな消えた！！


キーボードの
左上にある
Escキーを
押してみろ
Esc



機械語

0と1の組み合わせで意味を表す言語。直接この言語を使ってプログラムを作ることはほとんどない。

アセンブラ

機械語を簡単な単語に置き換えた言語。単純な命令しか用意されていないが、命令の処理速度は早い。

C/C++言語

現在最も一般的に使用されている言語。初心者には分かり難い部分もあるが、汎用性は高い。

BASIC言語

初心者にも分かり易いように設計されている言語。プログラムの間違えを見付け易く、修正も簡単に行える。

書いたぜ
PRINT 5+8■
PRINTっていうのは表示しろって命令だ
この命令は後ろに書いた5+8を計算して表示しろってことさ
スペースキーで空白をあけて5+8と書く
カチカチ
+はShiftキーを押しながら…と
Enterって書いてあるキーを押すんだ
ふむふむ…
それでどうやったら命令を聞いてくれるの？
カチッ
これだな
Enterキーが実行しろっていう意味のキーなんだ！！
PRINT 5+8
13
OK
■
おっ！計算されて13が表示された

PRINT 223-112
111
OK
■

ちゃんと命令が
実行されて答えが
表示されたぞ！！

同じように
引き算もやってみろ
-（マイナス）と
_（アンダーバー）を
間違えるな

ラジャー

カチャ

掛け算は＊（アスタリスク）

割り算は／（スラッシュ）
を使うんだ

PRINT 4＊(3+7)／3
13.3333

おおっ！ちゃんと
小数点付きで表示されてる

それに計算は括弧も
使えるんだ

どれどれ

まあゲームの場合
ほとんどの計算は
四則演算で足りる

これで四則演算は
計算できるな

へ！っ

10 PRINT 20 * 30■
ちゃちゃっとな
プログラムの入力
次はプログラムだ
さっきの命令の前に10を入れて何か計算させてみろ
ラジャー
Enterキーで実行っと…
あれ？答えを表示しなくなっちゃった…
同じ要領で20と30を付けた命令を書いてみるんだ
書いた後はそのつどEnterキーを押すんだ
命令の先頭に付けた数字のことを行番号っていうんだ
そして行番号が付いた命令はプログラムとして記録される
10 PRINT 20 * 30
20 PRINT 15 / 5
30 PRINT 100 / 3
■
やっぱり何もしなくなったよ…

LIST
10 PRINT 20 * 30
20 PRINT 15 / 5
30 PRINT 100 / 3
■
おっ！
さっき書いたものが
表示されたぞ
LISTって
打ち込んで
実行してみろ
LIST…
Enterキーと
LISTってのは
打ち込んだプログラムを
表示させる命令だ
つまり行番号が
付いたことで
こいつらはプログラム
として記録されたのさ
RUN
実行を開始します。
600
3
33.3333
おおっ
答えがいっぺんに
表示された！！
RUNと書いて
実行してみろ
RUN…で
Enterキー…と

LISTがプログラムの表示で
RUNがプログラムの
実行…と
覚えたぜ
RUNというのが
プログラムを実行
させる命令だ
プログラムを
行番号の小さい順に
実行したのさ
10 PRINT 20＊30
20 PRINT 1
30 PRINT 100
11～19の
行番号が
追加できる
10 PRINT 20＊30
:
20 PRINT 15 / 5
30 PRINT 100
行番号を10ずつ
増やすのは
プログラムを後で追加
したくなった時に間の
番号が使えるからだ
なるほど…
LIST
10 PRINT 20＊3
30 PRINT 100 / 3
おおっ 確かに！！
これで
20行目が
削除された
LIST表示、と
プログラムを1行
消したいときは
行番号だけ入力して
Enterキーだ
20と入れて
Enterキーで…
NEW …
実行っと
プログラムを全部
消したい場合は
NEW命令だ

了解！！
新しいプログラムを
書き始める前に
使えばいいんだ
間違えて使うなよ

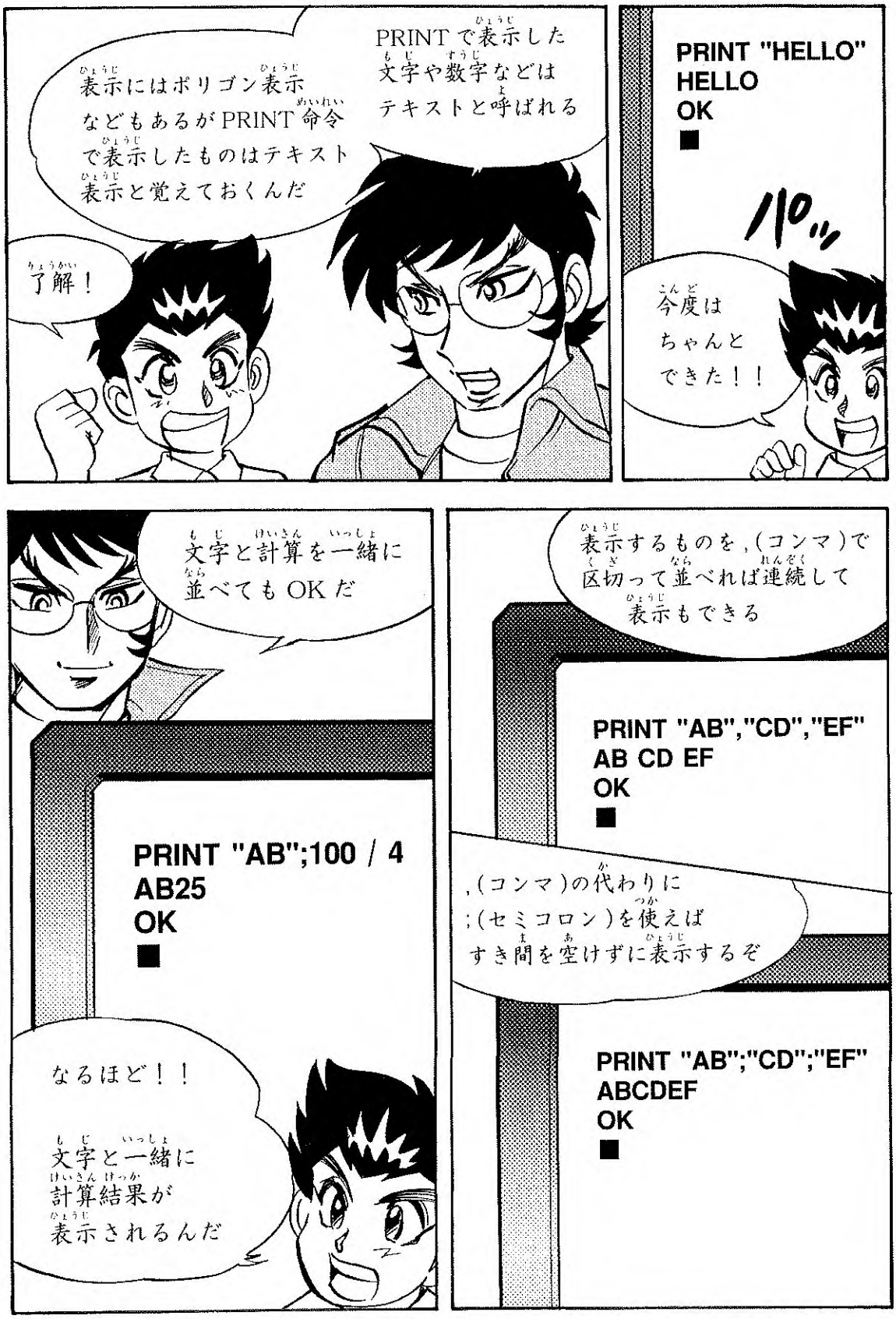
LIST
OK
■
あっ ほんとだ
全部消えちゃった

こうかな…
PRINT HELLO ■
カチャカチャ

PRINT HELLO
0
OK
■
へっ？
0が表示された
どういうことだ？

文字を表示させる
今度は文字を表示
させてみるか
HELLOって
表示させてみろ

PRINT "HELLO"
なるほど
文字を表示する場合は前後に
"（ダブルクォーテーション）
を付けるんだ


PRINTで表示した文字や数字などはテキストと呼ばれる
表示にはポリゴン表示などもあるがPRINT命令で表示したものはテキスト表示と覚えておくんだ
了解！
PRINT "HELLO"
HELLO
OK
■
110%
今度はちゃんとできた！！
文字と計算を一緒に並べてもOKだ
PRINT "AB";100 / 4
AB25
OK
■
なるほど！！
文字と一緒に計算結果が表示されるんだ
表示するものを,(コンマ)で区切って並べれば連続して表示もできる
PRINT "AB","CD","EF"
AB CD EF
OK
■
,(コンマ)の代わりに;(セミコロン)を使えばすき間を空けずに表示するぞ
PRINT "AB";"CD";"EF"
ABCDEF
OK
■

変数ってのは簡単に言えば
数字や文字を入れる入れ物だ
数字を入れる変数を
数値変数
文字を入れる変数を
文字変数と呼ぶ
ABCDE
0123
文字変数
数値変数
変数について
次は変数だ
変数？
A=10と打ち込んで
実行してみろ
カチャ
カチャ
A=10で
Enterキーと…
PRINT A
10
おお！！
10が表示された
変数Aに10が
入ってることが
確認できただろう
PRINT Aを
実行させてみろ
PRINT A…
Enter…っと
何も起きない
けど…
これはAという
数値変数に10を
入れたんだ

=(イコール)記号は右側のものを
左側の箱に入れろっていう意味だ

A

10

つまり A=10 てのは
数値 10 を変数 A に
代入しろってことだ

変数っていうのはこういう
ふうに数字や文字を入れ
ておく箱みたいなものだ

変数

プログラミングに変数は
欠かせないものだから
しっかりおぼえるように

PRINT 命令で
確認だ

B$="ABC"…で
Enter…と

ところで文字の場合は
どうすればいいの？

B$="ABC" と書いて
実行してみな

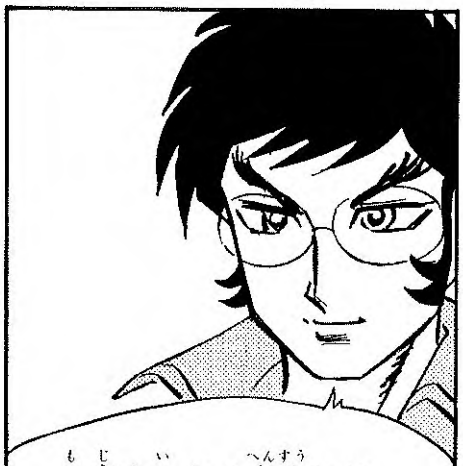
おし！！
ちゃんとできた
みたいだ

```
B$="ABC"
PRINT B$
ABC
OK
■
```

変数名の最後が$記号の場合は文字列変数として扱われるんだ

B$ = "ABC"

そして入れる文字は前後を"(ダブルクォーテーション)で囲んでやるわけだ



文字を入れる変数には変数名の後ろに$(ドル)記号を付けるんだ

"ABC"

"(ダブルクォーテーション)で囲まれた部分は文字列と呼ぶ

○ SCORE PT_HIT TEXT9$

× NOTE# 51D

変数名に数字は使えるが記号は_(アンダーバー)だけ

それから変数名の先頭に数字は使えない

だいたいわかったぞ

数値変数に文字列を入れたりはできないから気を付けろ

変数名は$記号も含めて10文字まで使える

これを実行すると
どうなる？

```
10 A=7+8
```

```
10 A=7+8
20 B=A-3
30 C=A+B
40 PRINT C
```

※答えを見る前に挑戦してみよう

LIST
10 INPUT A
20 PRINT A
RUN
実行を開始します。
?
■
あれ？
"?"が出て
止まっちゃった
えーと
LISTで
確認して
から…
RUNで
実行っと
ぽちぽち
キーボードから値を入れる
次はキーボードから
変数に値を入れる命令だ
10 INPUT A
20 PRINT A
このプログラムを
実行してごらん
10 INPUT A
20 PRINT A
RUN
実行を開始します。
?
123
123
おっ！！
何か数字を入れて
Enterキーを押すんだ
じゃ適当に
123と入れて
Enter…と
十千万
そうさ
INPUTってのはキーボード
から打ち込まれた値を変数に
代入しろっていう命令なんだ
打ち込んだ数字が
そのまま表示された

その場合はINPUT命令の次にPROMPT命令を付ける

PROMPT命令の後ろに表示する文字列を書いたら:(コロン)で区切ってから変数名だ

```
10 INPUT PROMPT "How much?":A
RUN
実行を開始します。
How much?
■
```

なるほど
PROMPTってのが文字を表示させる命令なんだ
PRINTみたいなものだな

さっきのは入力した値をAに入れてその値を表示するプログラムだ

入力待ちのときに?(クエスチョン)以外の文字を表示することもできる

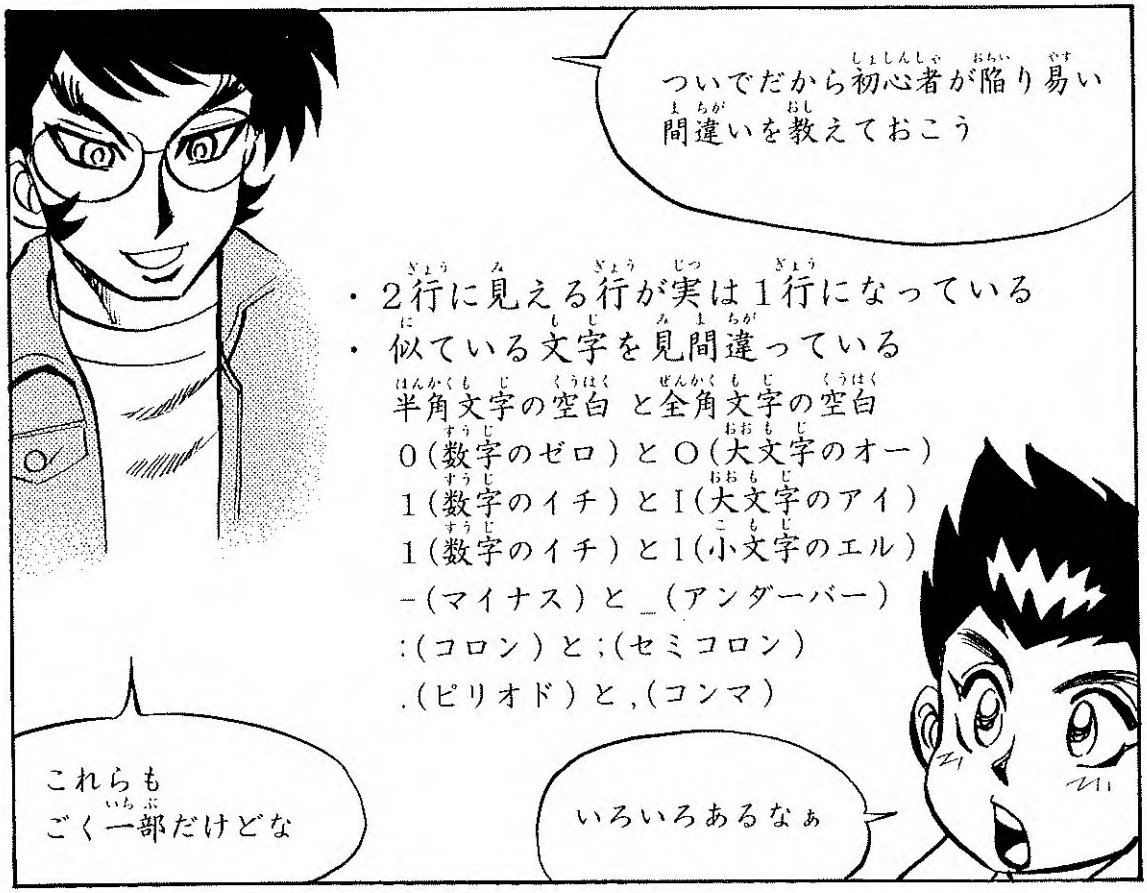
了解！！

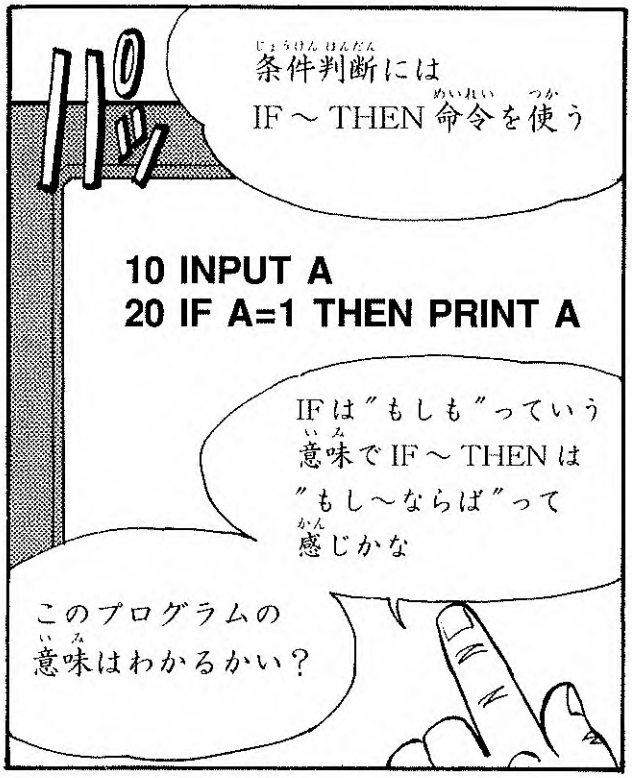
代入する変数には文字列変数も使える

文字列を入力するときは"(ダブルクォーテーション)を付ける必要はないぞ

えーと
こんな感じ
かな…
10 INPUT A
20 INPUT B
30 PRIMT A+B
RUN ■
カタカタ
RUNで実行…と
次はINPUTを使って
足し算をするプログラム
を作ってごらん
これはエラーメッセージで
プログラムに間違いがある
ってことだ
エラーメッセージの
前に間違っている行が
表示されてるだろ
10 INPUT A
20 INPUT B
30 PRIMT A+B
RUN
30 PRIMT A+B
構文の誤りです。
■
ん…?
実行されずに
止まったぞ…
LISTで
プログラムを
見直して…
間違いを直して
もう一度実行だ
カーソルを移動して
MをNに変えて
Enterキー…と
カタカタ
えーと…あっ!
PRINTをPRIMT
と間違って書いて
ある
RUN
30 PRIMT A+B
構文の誤りです。

プログラムを書くと間違えることもあるからな
そんなときはめげずにエラーメッセージをよく読んで修正だ
了解！！
20 INPUT B
30 PRINT A+B
RUN
実行を開始します
?
123
?
234
357
今度はバッチリできたぞ！！
LIST
10 INPUT A
20 INPUT B
30 PRINT A+B
RUN■
もう間違いは無いようだな…
もう一度 RUN だ
カチッ
Enter


ついでだから初心者が陥り易い間違いを教えておこう
・2行に見える行が実は1行になっている
・似ている文字を見間違っている
半角文字の空白 と全角文字の空白
0(数字のゼロ)と O(大文字のオー)
1(数字のイチ)と I(大文字のアイ)
1(数字のイチ)と l(小文字のエル)
-(マイナス)と _(アンダーバー)
:(コロン)と ;(セミコロン)
.(ピリオド)と ,(コンマ)
これらもごく一部だけどな
いろいろあるなぁ



比較に使う記号

=	左側と右側が同じ
>	左側が右側よりも大きい
<	左側が右側よりも小さい
>=	左側が右側と同じか大きい
<=	左側が右側と同じか小さい
<>	左側と右側が違う

入力されたAが0より大きくてなおかつBが10より小さい場合にOKを表示…かな

えーと

比較の条件をさらに重ねる場合はこうなる

```
10 INPUT A
20 INPUT B
30 IF A>0 AND B<10 THEN PRINT "OK"
```

ANDは〝なおかつ〟っていう意味だ
このプログラムの内容がわかるか？

正解だ！！
違う意味で条件を重ねることもできる

AND　なおかつ
OR　もしくは
NOT　～でないならば

それから
条件に合わなかったときに命令を実行させたい場合はELSEを使って命令を追加することもできる

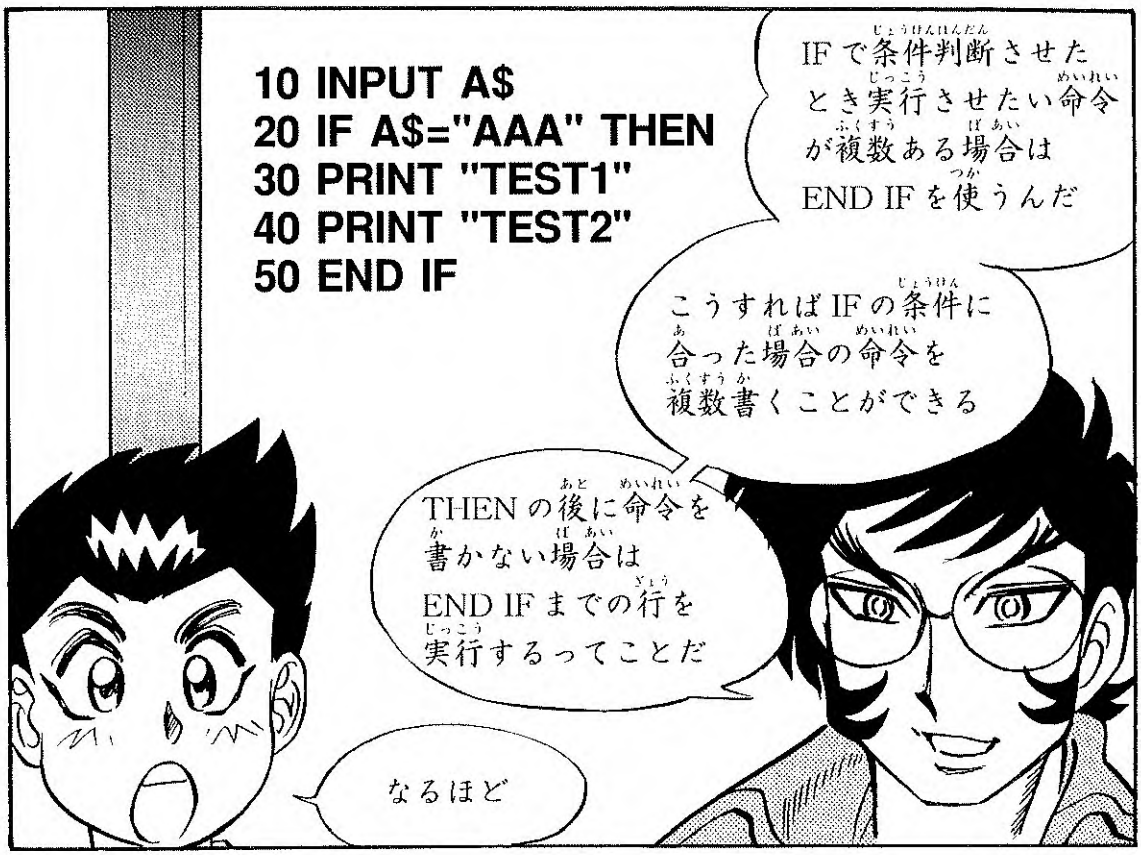
```
10 INPUT A$
20 INPUT B$
30 IF A$="AAA" OR B$="BBB" THEN
 PRINT "Hit" ELSE PRINT "Other"
```

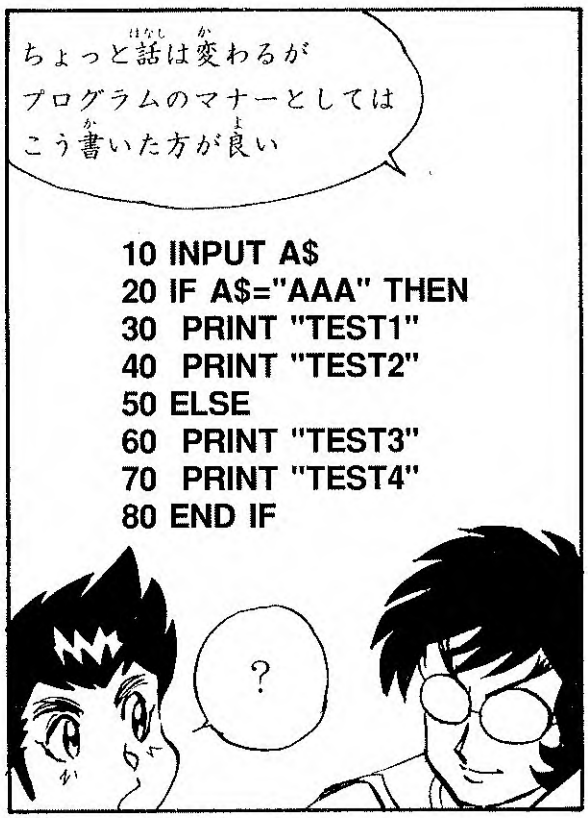
あれ？

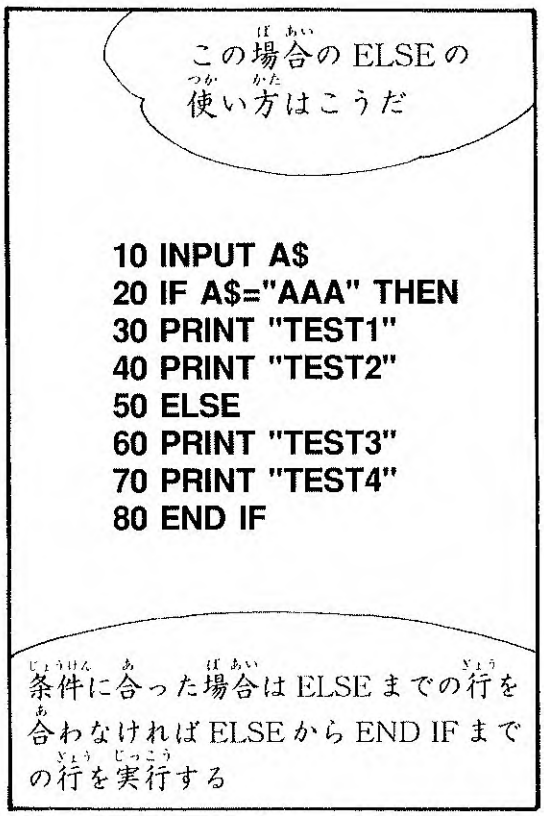
文字列変数を比較することもできる

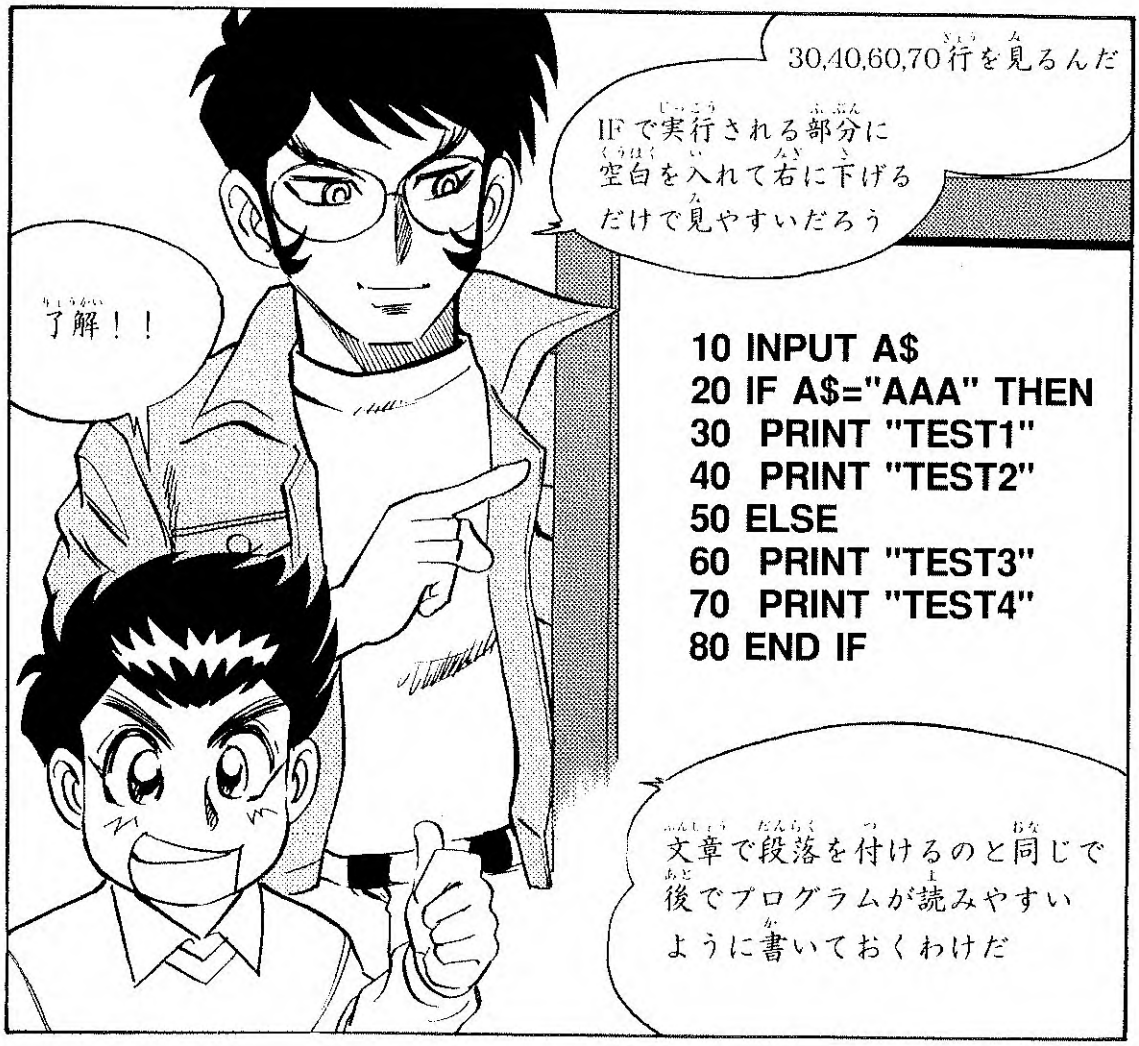
```
IF A$="AAA"
```

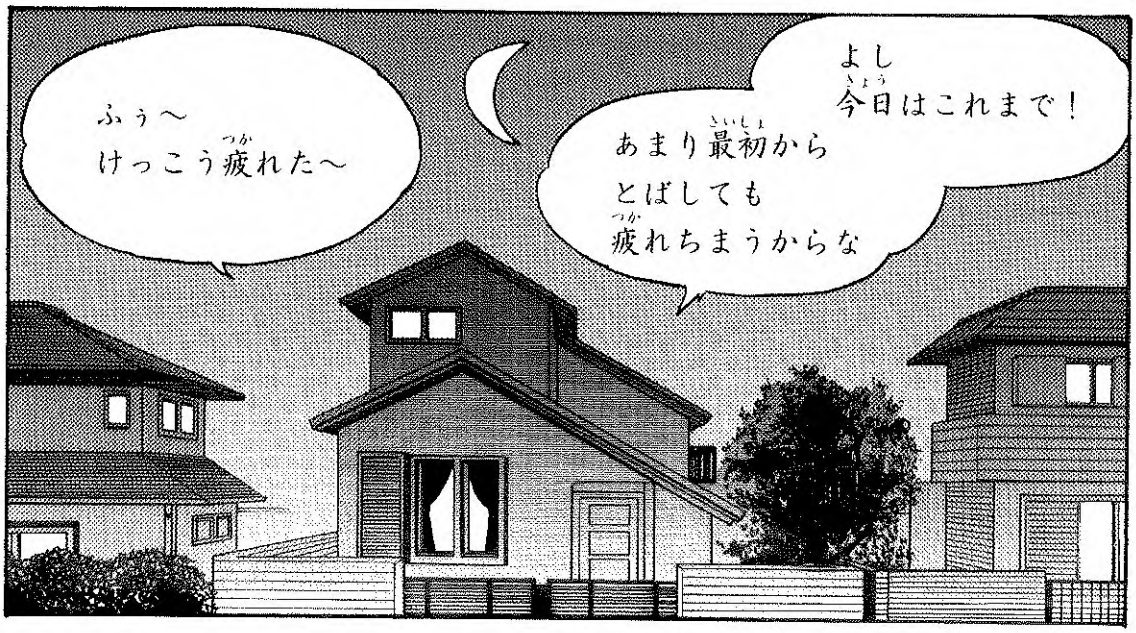
ああ
それは大丈夫だ
1行に入りきらなくて
2行目にはみ出してるよ
10 INPUT A$
20 INPUT B$
30 IF A$="AAA" O
PRINT "Hit" ELSE
プログラムの1行の文字数は
行番号を含めて132文字まで
画面だと約3行まで大丈夫だ
10 INPUT A$
20 INPUT B$
30 IF A$="AAA" OR B$="BBB" THEN
PRINT "Hit" ELSE PRINT "Other"
この例はA$がAAAもしくは
B$がBBBの文字列のときにHitを
そうでないときはOtherを表示する
プログラムだ
おう！！
条件判断はプログラムのキモだ
しっかり覚えて使いこなせる
ようになれよ


10 INPUT A$
20 IF A$="AAA" THEN
30 PRINT "TEST1"
40 PRINT "TEST2"
50 END IF
IFで条件判断させたとき実行させたい命令が複数ある場合はEND IFを使うんだ
こうすればIFの条件に合った場合の命令を複数書くことができる
THENの後に命令を書かない場合はEND IFまでの行を実行するってことだ
なるほど


ちょっと話は変わるがプログラムのマナーとしてはこう書いた方が良い
10 INPUT A$
20 IF A$="AAA" THEN
30 PRINT "TEST1"
40 PRINT "TEST2"
50 ELSE
60 PRINT "TEST3"
70 PRINT "TEST4"
80 END IF
?


この場合のELSEの使い方はこうだ
10 INPUT A$
20 IF A$="AAA" THEN
30 PRINT "TEST1"
40 PRINT "TEST2"
50 ELSE
60 PRINT "TEST3"
70 PRINT "TEST4"
80 END IF
条件に合った場合はELSEまでの行を合わなければELSEからEND IFまでの行を実行する


30,40,60,70行を見るんだ
IFで実行される部分に
空白を入れて右に下げる
だけで見やすいだろう
了解！！
10 INPUT A$
20 IF A$="AAA" THEN
30 PRINT "TEST1"
40 PRINT "TEST2"
50 ELSE
60 PRINT "TEST3"
70 PRINT "TEST4"
80 END IF
文章で段落を付けるのと同じで
後でプログラムが読みやすい
ように書いておくわけだ


よし
今日はこれまで！
あまり最初から
とばしても
疲れちまうからな
ふぅ〜
けっこう疲れた〜

コラム1 日本語を入力してみよう

"EZ-BASIC"では、日本語を使うことができます。

ソフトウェアキーボードを使う

コントローラを操作してソフトウェアキーボードから日本語を入力することができます。

◆日本語入力モード切り替え

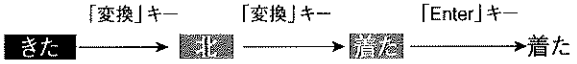
・「漢」キーを押すと日本語入力モードに切り替わります。
※再び「漢」キーを押すと日本語入力モードを解除します。

◆文字の入力

・方向キーで文字を選び、○ボタンで文字を入力します。入力した文字は漢字変換の候補として反転表示されます。
・「Shift」キーを一度押すと「ゃ」「ゅ」「ょ」「っ」などの文字や記号を入力することができます。
※再び「Shift」キーを押すとShift状態が解除されます。

◆漢字変換

・「変換」キーを押すと、入力した文字を漢字に変換します。
「変換」キーを押す度に、次の漢字候補が表示されます。
「Shift」キーを押すと、「変換」キーが「前候補」キーになります。このキーを押すとひとつ前の漢字候補が表示されます。
・変換した漢字でよければ「Enter」キーを押します。



・変換中に、「無変換」キーまたは「ESC」キーを押すと変換を解除します。

◆入力モードの変更

・ひらがな入力モード‥「かな」キーを押します。ひらがなの入力および漢字変換ができます。画面左下に「あかナ」と表示されます。
・カタカナ入力モード‥Shift状態で「カナ」キーを押します。カタカナを入力できます。画面左下に「アかナ」と表示されます。カタカナ入力は漢字変換できません。
・アルファベットモード‥「英」キーを押します。全角英字を入力できます。画面左下に「A」と表示されます。再度「英」キーを押すと解除できます。

◆その他のコントローラ操作

・L1ボタン+方向キー‥‥ソフトウェアキーボードの移動
・L2ボタン+方向キー‥‥カーソルの移動
・R1ボタン‥‥ShiftキーのON/OFF切り替え
・R2ボタン‥‥「Enter」キーと同等の機能

USBキーボードを使う

"PlayStation 2"(本体)のΨ(USB)端子にUSB接続のキーボードをつなぐと、USBキーボードから日本語を入力することができます。
※USBキーボード接続時、ソフトウェアキーボードからの入力はできなくなります。

◆日本語入力モード切り替え

・半角／全角キーを押すと日本語入力モードになります。
※再び半角／全角キーを押すと日本語入力モードを解除します。

◆文字の入力

・キーボードから直接文字を入力します。入力した文字は漢字変換の候補として反転表示されます。

コラム1 日本語を入力してみよう
"EZ-BASIC"では、日本語を使うことができます。

ローマ字入力の場合、キートップに表記されているアルファベットが入力されます。カナ入力の場合、キートップに表記されている日本語が入力されます。

・ESCキーを押すと、入力文字を消去します。

◆漢字変換

・変換（次候補）キーを押すと、入力した文字を漢字に変換します。
変換（次候補）キーを押す度に、次の漢字候補が表示されます。
Shiftキーを押しながら変換（次候補）キーを押す度に、ひとつ前の漢字候補を表示します。

・変換した漢字でよければEnterキーを押します。

・無変換キーまたはESCキーを押すと、変換を解除します。

◆カナ入力／ローマ字入力の切り替え

・Altキーを押しながらカタカナ／ひらがなキーを押すことで、カナ入力とローマ字入力を切り替えます。

◆入力モードの変更

・ひらがな入力モード … カタカナ／ひらがなキーを押します。ひらがなの入力および漢字変換ができます。画面左下に「あカナ」または「あR」と表示されます。

・カタカナ入力モード … Shiftキーを押しながらカタカナ／ひらがなキーを押します。カタカナを入力できます。画面左下に「アカナ」または「アR」と表示されます。入力したカタカナは漢字変換できません。

・アルファベットモード … CapLock／英数キーを押します。全角英数字や一部の記号を入力できます。画面左下に「A」と表示されます。

◆変換部分の変更

・漢字変換中、左右のカーソルキーで変換する語を切り替えます。

右カーソルキー

・漢字変換中、Shiftキーを押しながら左右のカーソルキーを押すと、変換する文字列を1文字ずつ調節することができます。

Shift+左カーソルキー

◆促音・拗音の入力

・ローマ字入力

「ゃ」「ゅ」「ょ」「っ」などの文字は次のように入力します。

かったー	katta-
きゃんぷ	kyanpu または kyannpu
じぇっと	jetto
しゃっくり	syakkuri

・カナ入力

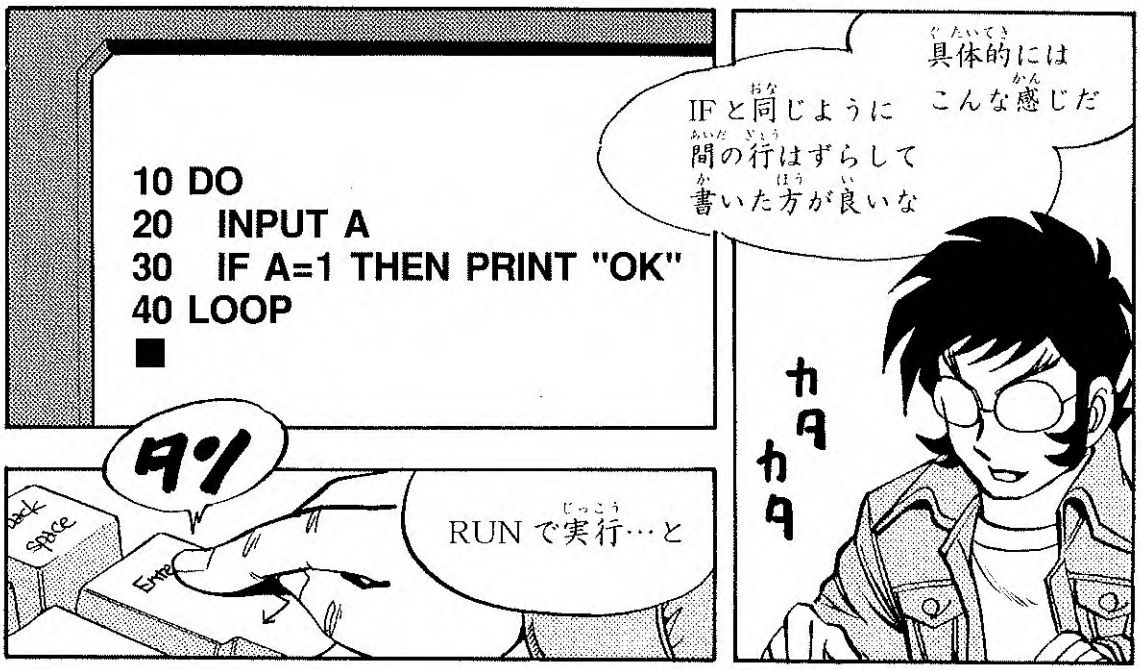
かったー	か　Shift+つ　た　ー
きゃんぷ	き　Shift+や　ん　ふ　゜
じぇっと	し　゛　Shift+え　Shift+つ　と
しゃっくり	し　Shift+や　Shift+つ　く　り

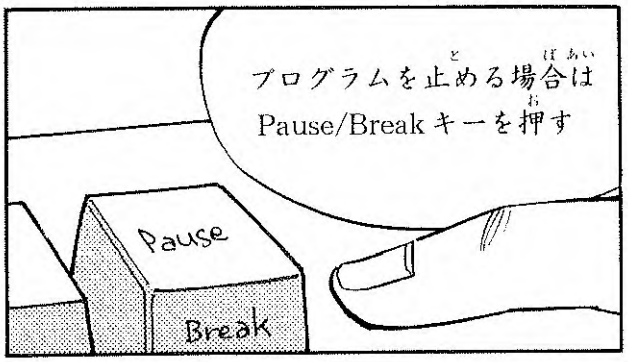
簡単なゲームを作ろう

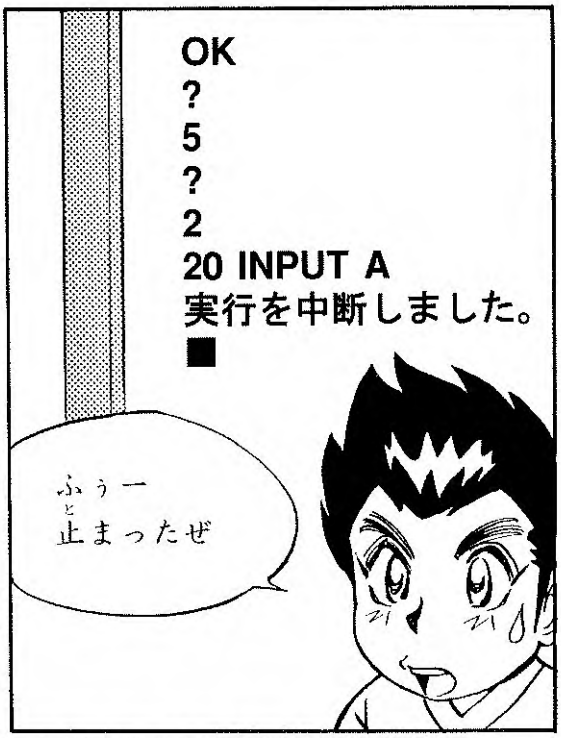
今日から本格的にプログラムのことを教えて行くぞ
よっ！
ガラッ
ピーンポーン
またヨロシク頼ンます！！
繰り返す方法？
まずはプログラムを繰り返す方法を教えよう

ふんふん
ゲームってのは普通プログラムが終了することなく繰り返し動いているだろう

DO
LOOP
繰り返しのことをループといって命令はDOとLOOPを使うんだ
そうするとDOからLOOPまでの行が繰り返し実行される


10 DO
20 INPUT A
30 IF A=1 THEN PRINT "OK"
40 LOOP
■
具体的にはこんな感じだ
IFと同じように間の行はずらして書いた方が良いな
カタカタ
ﾀﾝ
back space
Enter
RUNで実行…と





※アナログコントローラ(DUALSHOCK 2)のSTARTボタン＋SELECTボタンでも実行を中断できます。

具体的にはこんな感じだ
10 DO
20 INPUT A
30 IF A=1 THEN PRINT
40 LOOP UNTIL A>10
今のDO-LOOP命令には
繰り返しを終了する条件を
書かなかったからな
DO-LOOP命令には
繰り返しを終了する
条件を付けられるんだ
DOかLOOPの後に
UNTILを付けることで
終了する条件が書ける
んだ
この例だと
Aが10を越えたら
DOに戻らず終了するんだ
なるほど…
IFみたいに条件判断に
ANDとかORは使えるの？
良い質問だ
まったく同じように使えるぞ

```
10 DO
20  INPUT A
30  IF A=1 THEN PRINT "OK"
40 LOOP WHILE A<=10
```

このプログラムは
前のUNTILとまったく
同じ動きをする

Aと10を比較する
条件が変わってること
に注目だ

UNTILと同じような
命令でWHILEって
のもある

これは終了するための
条件ではなく
繰り返すための条件を
付けて使うんだ

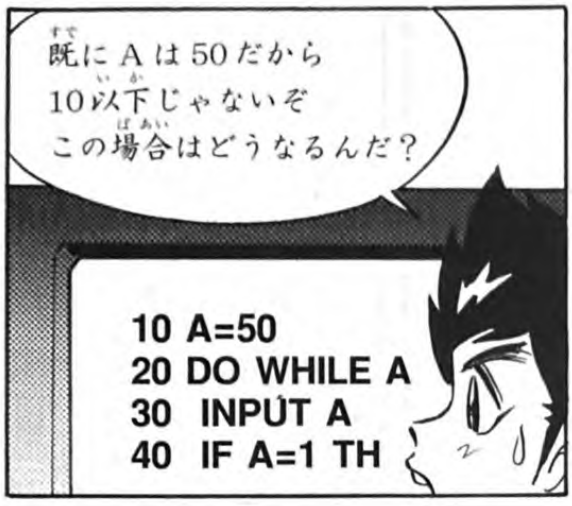
それじゃ
これはどうなる？

```
10 A=50
20 DO WHILE A<=10
30  INPUT A
40  IF A=1 THEN PR
50 LOOP
```

※答えを見る前に挑戦してみよう

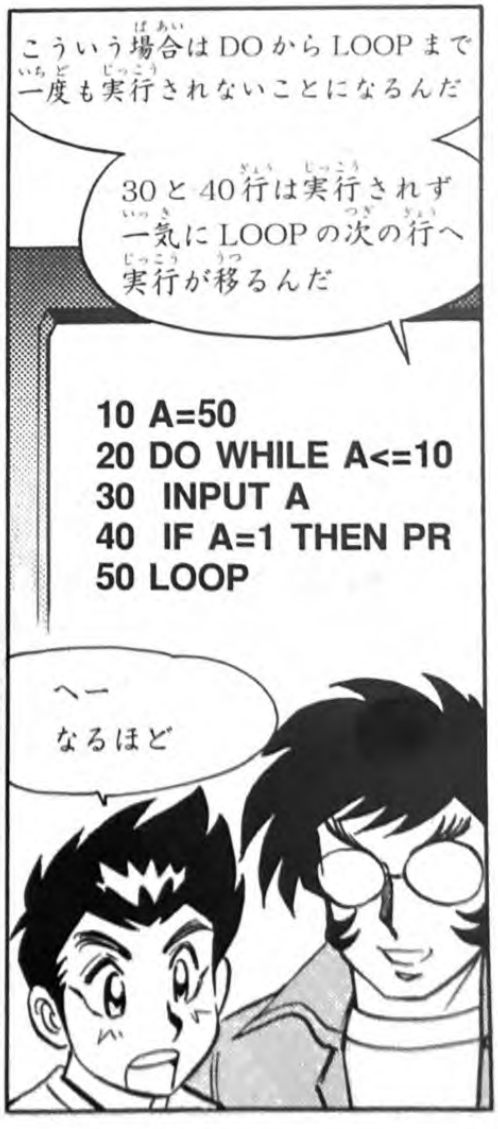
終わるための条件を反対に
すれば繰り返すための
条件と同じってことか

そうだ
条件が反対なだけで
UNTILとWHILEは
ほぼ同じ命令なんだ


既にAは50だから
10以下じゃないぞ
この場合はどうなるんだ？
10 A=50
20 DO WHILE A
30 INPUT A
40 IF A=1 TH

えーと
Aに50を入れて
次にAが10以下のときに
DOからLOOPまでを
繰り返すんだから…
ん…？

DO-LOOPによる繰り返しは
プログラムの基本だからな
バッチリ理解しておけよ


こういう場合はDOからLOOPまで
一度も実行されないことになるんだ
30と40行は実行されず
一気にLOOPの次の行へ
実行が移るんだ
10 A=50
20 DO WHILE A<=10
30 INPUT A
40 IF A=1 THEN PR
50 LOOP
へー
なるほど

繰り返し途中で
ループから抜ける
方法も教えておこう
EXIT DOを使うんだ
DOから脱出ってことだ

こんな感じだ
この例だと
10行でAに1を入れて
20行のDOは繰り返しの
始まりだ
10 A=1
20 DO
30 A=A+1
40 IF A=10 THEN EXIT DO
50 LOOP
60 PRINT A
■
30行でAに1を足して
40行でもしAが10になったら
EXIT DO命令が実行されて
LOOPの次の行に抜け出す
Aが10でなければ抜け出さないので
またDOのある20行に戻る…
なるほど
だから繰り返しを抜けて
60行で表示されるAは
10になる

良いところに気が付いたな
この例ではそうだ
プログラムを書いていると
ループを途中で抜けたい場合が出てきたりするので
そういうときに使うんだ
了解！
でもこれってUNTILやWHILEに条件を付ければ済むんじゃ…
DO-LOOPと同じような繰り返しをさせる命令には
FOR-NEXTってのもある
これは変数を使ってカウントしながらループする
DO-LOOPのようなものだ
30行のNEXT命令で
Aに1が足されて
FORに戻るんだ
この例を見てみろ
10 FOR A=1 TO 5
20 PRINT A
30 NEXT A
これは変数Aに
最初1を入れてTOの後に
書かれた5まで繰り返す
ってことだ
繰り返す内容は20行に
書かれてる変数Aの
表示だな

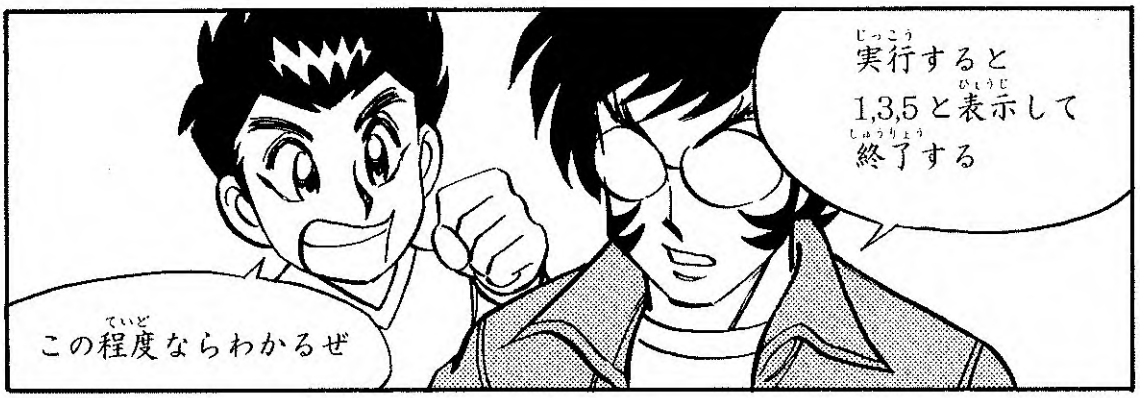
10 A=1
20 DO UNTIL A>5
30 PRINT A
40 A=A+1
50 LOOP
このプログラムは前のFOR-NEXTとまったく同じ動作をする
これならDO-LOOPでも同じことはできるよ
そうだなDO-LOOPを使った場合と見比べてみよう
じゃあFOR-NEXT命令なんて必要無いじゃんか
まあそうとも言えるがちょっと行数が少なくなったろう？
ふーんそうなのかぁ…
プログラムをループさせたい場合変数を用意してカウントすることが多いんだ
そんなときはFOR-NEXTで書けば短くて分かりやすくなるってことさ

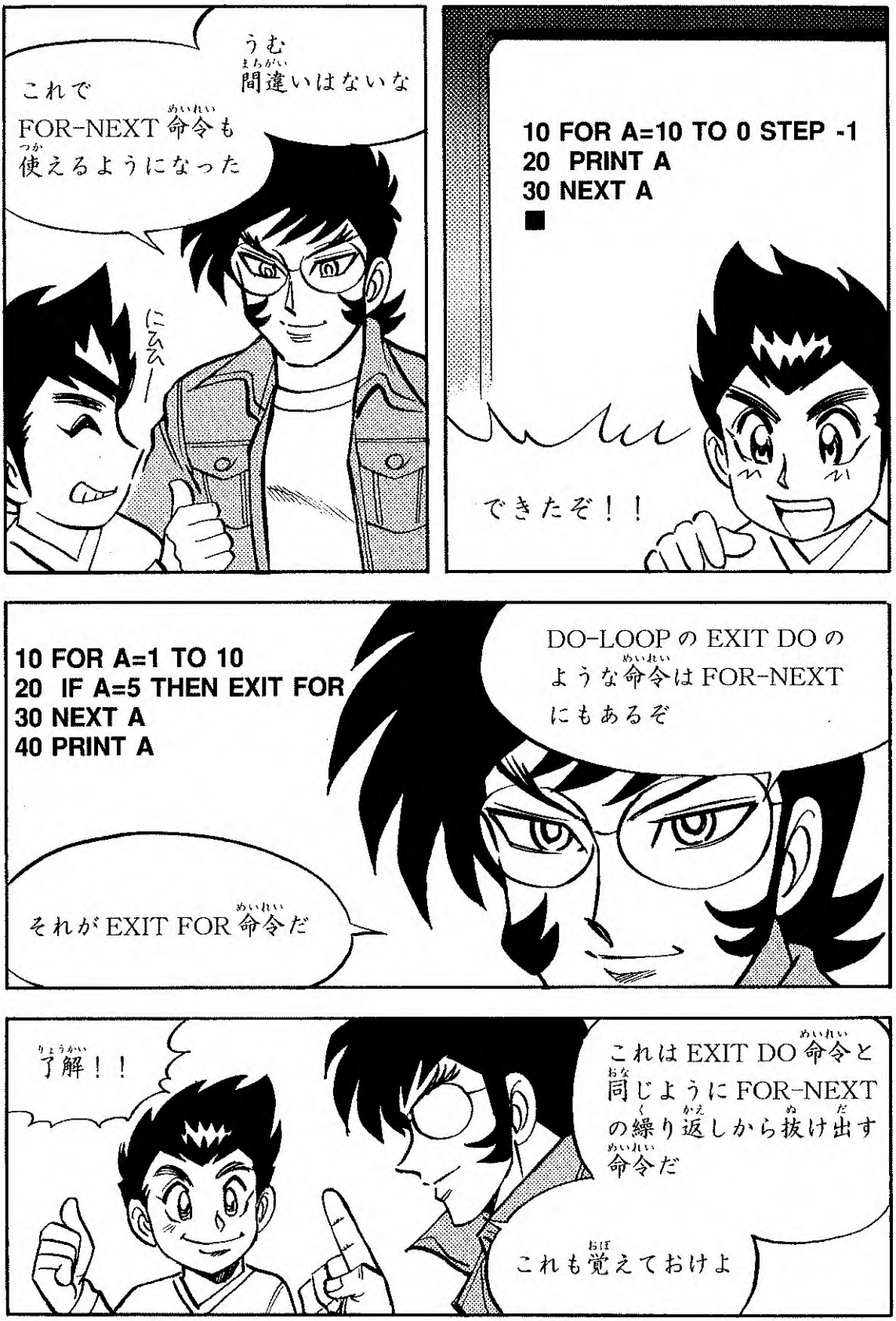
さっき30行のNEXT Aで
自動的に1が足されていたが
STEPを使うとそれを変える
ことができる

```
10 FOR A=1 TO 5 STEP 2
20  PRINT A
30 NEXT A
```

この例だと
NEXT命令が実行される
とき変数Aには2が
足されるようになる

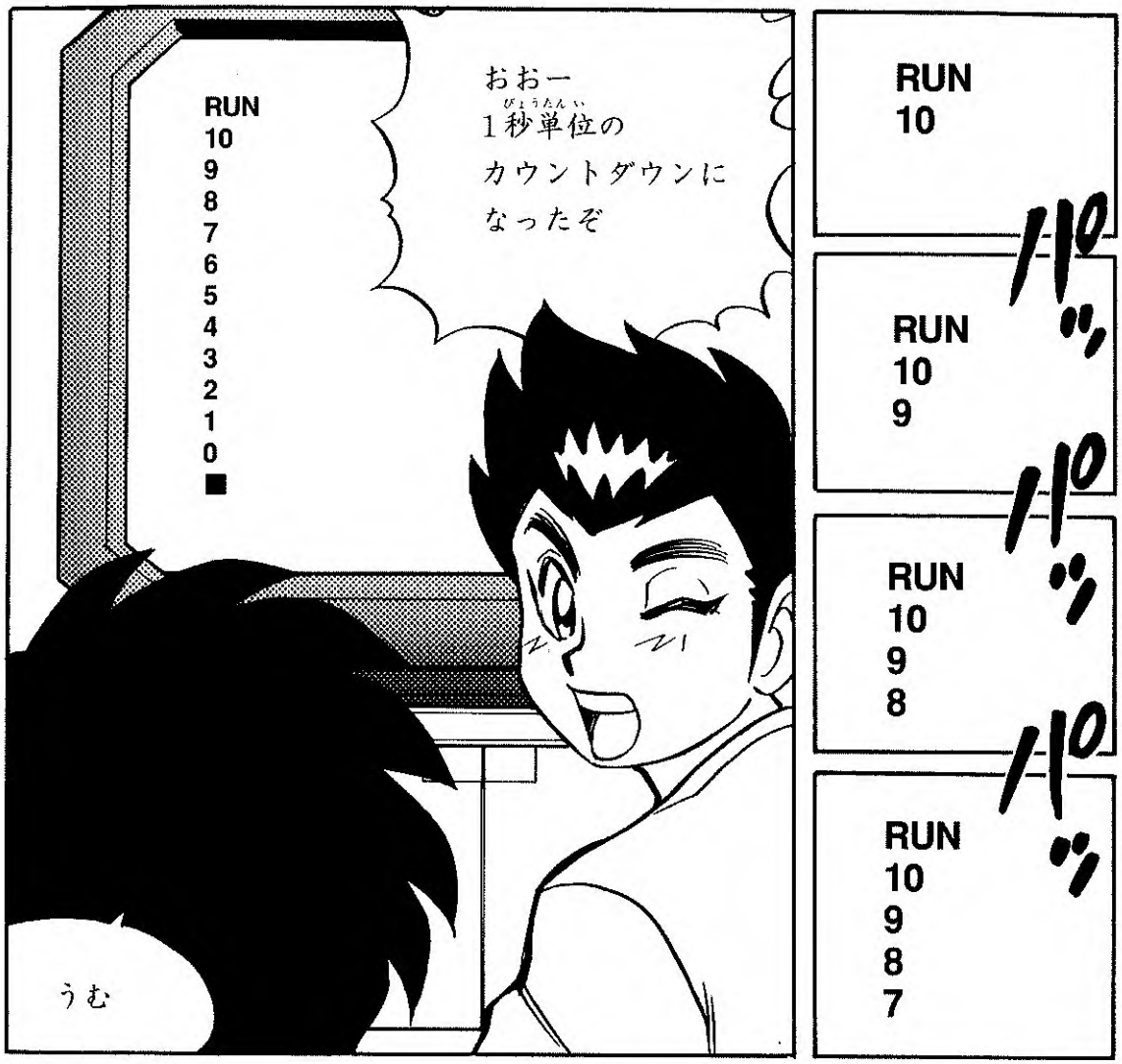
FOR-NEXT命令

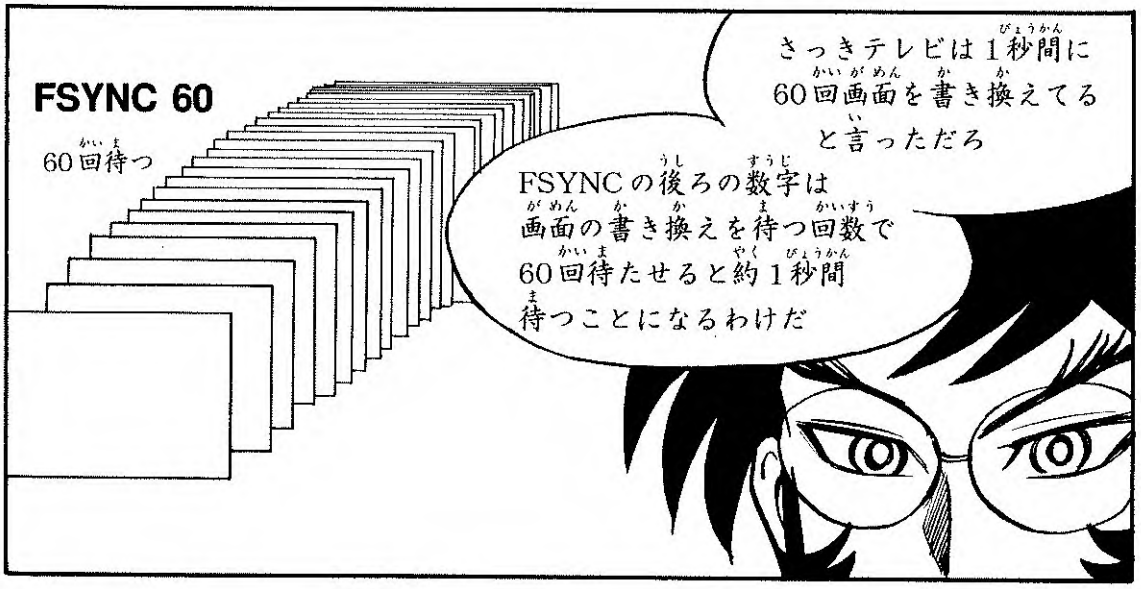



うむ
間違いはないな
これで
FOR-NEXT命令も
使えるようになった
にひひー
10 FOR A=10 TO 0 STEP -1
20 PRINT A
30 NEXT A
■
できたぞ！！
10 FOR A=1 TO 10
20 IF A=5 THEN EXIT FOR
30 NEXT A
40 PRINT A
DO-LOOPのEXIT DOのような命令はFOR-NEXTにもあるぞ
それがEXIT FOR命令だ
了解！！
これはEXIT DO命令と同じようにFOR-NEXTの繰り返しから抜け出す命令だ
これも覚えておけよ

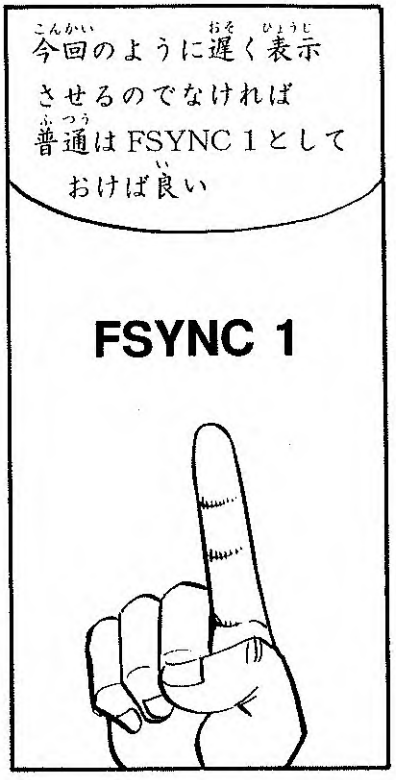
プログラムを実行したとき
表示が速すぎて見えない
場合がある
例えばさっきのカウントダウン
プログラムも速すぎて
カウントダウンぼくないだろ？
ちょっと気分転換に
簡単な命令を教えよう
RUN
10
9
8
7
6
5
確かに同時に表示
されてるみたいだった
ちょっと難しい説明だが
日本のテレビは1秒間に
60回画面を書き換えて
表示してるんだ
つまりゲームの方で1秒間に
60回以上の速さで画面を
書き換えてもテレビでは上手く
表示できないんだ

そこでテレビの表示とタイミングを合わせるFSYNC命令を使うんだ
RUN
RUN
10
9
さっきのプログラムだとテレビが1回画面を書き換える間にカウントダウンが全て終わっているんだ
それで同時に表示されてるように見えるのか
なるほど
FSYNC命令を実行するとプログラムが一時停止してテレビの表示タイミングに合わせてくれるんだ
ほいっ
RUN…と
10 FOR A=10 TO 0 S
20 PRINT A
25 FSYNC 60
30 NEXT A
■
これで実行してごらん
カウントダウンプログラムに1行追加しよう
チャ
チャッ


RUN
10
9
8
7
6
5
4
3
2
1
0
■
おおー
1秒単位の
カウントダウンに
なったぞ
うむ
RUN
10
RUN
10
9
RUN
10
9
8
RUN
10
9
8
7


FSYNC 60
60回待つ
さっきテレビは1秒間に
60回画面を書き換えてる
と言っただろ
FSYNCの後ろの数字は
画面の書き換えを待つ回数で
60回待たせると約1秒間
待つことになるわけだ

ゲームではメインループと
呼ばれる繰り返しの中で
テレビの表示とタイミングを
取るために実行するんだ
ふーん
なるほど…


今回のように遅く表示
させるのでなければ
普通はFSYNC 1として
おけば良い
FSYNC 1

プログラムで言うところの
関数ってのは
何か数字を渡すと別の数字を
返してくる変数みたいな
ものだ
数字
関数
別の数字
じゃあ
これ

INT関数
次は関数を
教えておこう
関数？

この関数は小数点以下を切り捨てた整数を返してくれるんだ
PRINT INT(3.5)
3
■
ハ°ッ
3.5が3になって返されたってことか
例えばINTという関数を使ってみよう
INT(3.5)
引数
そうだ関数を使う場合は渡す数字を括弧で囲んでやるんだ
括弧で囲われた渡す数字のことを引数と呼ぶ
例えばREPEAT$関数だこれは文字列を繰り返してつなげたものを返すんだ
繰り返しつなげる？
返してくる値が数字ではなく文字列の関数もあるこれは文字列関数って呼ぶ
おしえてー

実行するとこうなる

```
PRINT REPEAT$("GO!",5)
GO!GO!GO!GO!GO!
■
```

なるほど
GO!っていう文字が
5回つながって表示
されてる

この関数は引数が
2つ必要で
繰り返す文字列と
繰り返す回数を
渡すんだ

```
A=INT(3.5)
IF A=INT(3.5) THEN …

A$=REPEAT$("GO!",5)
IF A$=REPEAT$("GO!",5) THEN
```

関数が返す数字や文字列は
そのまま別の変数に代入したり
条件判断に使ったりと
変数と同じように扱える

まあ間違えて使っても
エラーが表示されて
教えてくれるから
そのときは別の名前を
付ければ良い

そっか

それと既に関数で使われてる
名前は変数名として使えない
ので注意だ

```
× INT=1
○ INTERNET=1

× REPEAT$="ABC"
○ REP$="ABC"
```

さっきのREPEAT$や
INTっていう名前は関数名
で使われてるから変数名には
できないのさ

この関数は0から1までのランダムな数字を返すんだ
ランダムってのはでたらめってことだ
PRINT RND()
0.448599
OK
■
何か訳のわからない数字が表示されたぞ
関数には引数が必要と言ったけど引数の無い関数もある
カタカタ
例えばRND関数には引数が無い
でたらめな数字を返すから実行するたびに違った値を返してくるのさ
PRINT RND()
0.448599
OK
PRINT RND
0.51686
OK
おーっ確かに違う数字が表示された
それと引数のいらない関数は括弧を付けなくても大丈夫だ
ﾀﾀﾀﾀｯ

RNDが返した数字を6倍すれば
0から6未満の数字になる

それをINT関数の引数にすれば
小数点以下が切り捨てられるから
0から5までの数字になる…

INT関数で

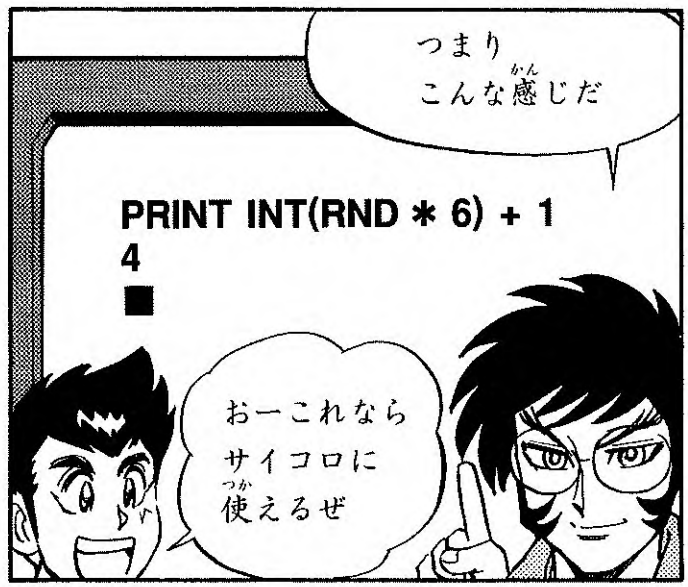
$0.0 \times 6 = 0.0 \rightarrow 0$

⋮

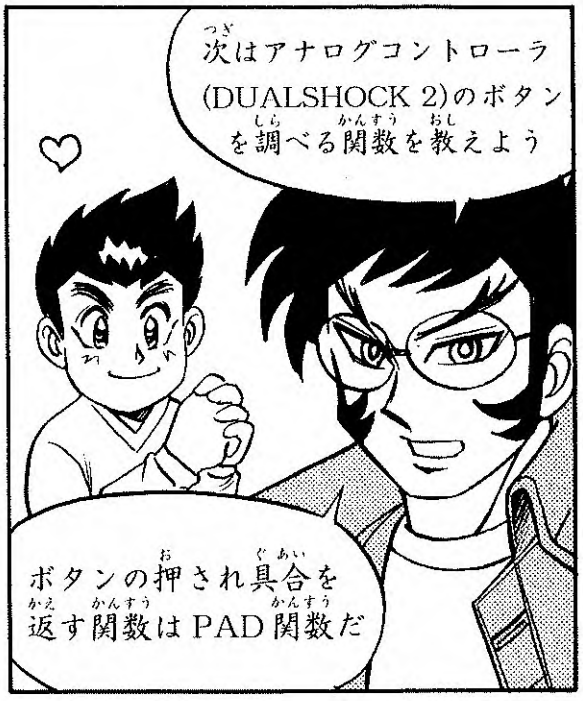
$0.2 \times 6 = 1.2 \rightarrow 1$

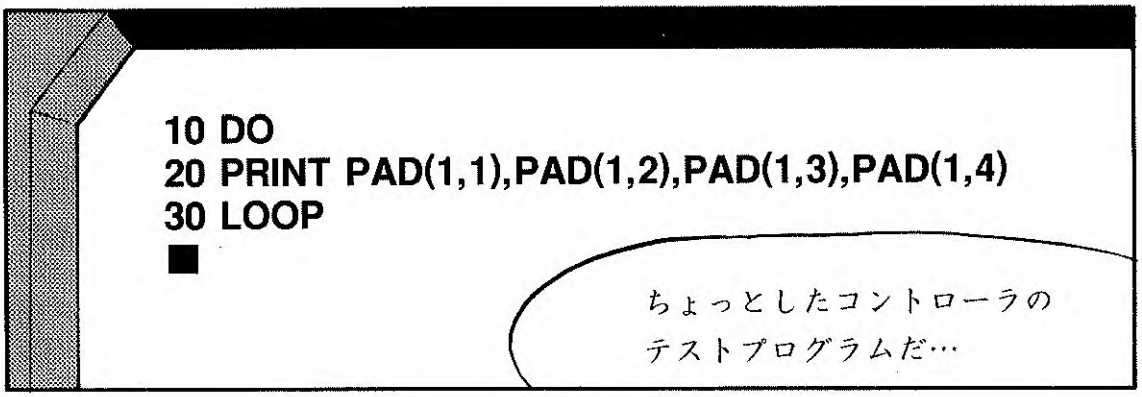
⋮

$0.999 \times 6 = 5.999 \rightarrow 5$

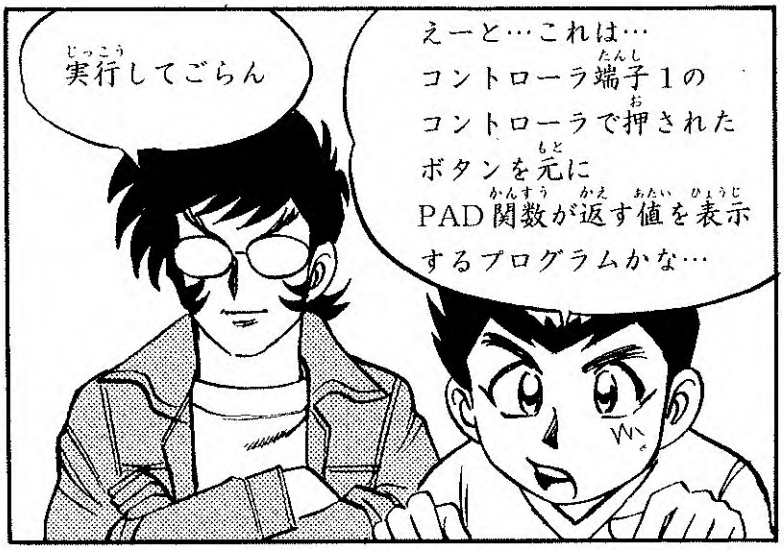


引数は２つで
調べるコントローラ端子の
番号と各ボタンの番号だ
コントローラ端子の番号は
１か２でいいとして
各ボタンの番号が何番なのかは
付属の Reference Manual を見てくれ


次はアナログコントローラ
(DUALSHOCK 2)のボタン
を調べる関数を教えよう
ボタンの押され具合を
返す関数は PAD 関数だ

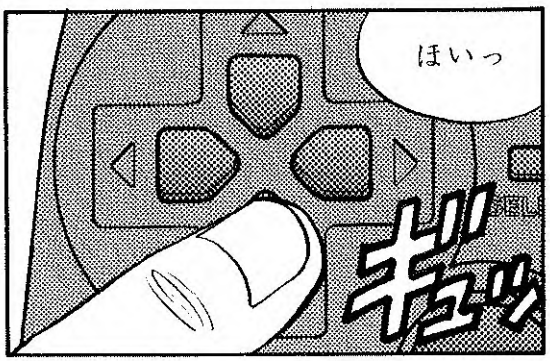

10 DO
20 PRINT PAD(1,1),PAD(1,2),PAD(1,3),PAD(1,4)
30 LOOP
■
ちょっとしたコントローラの
テストプログラムだ…

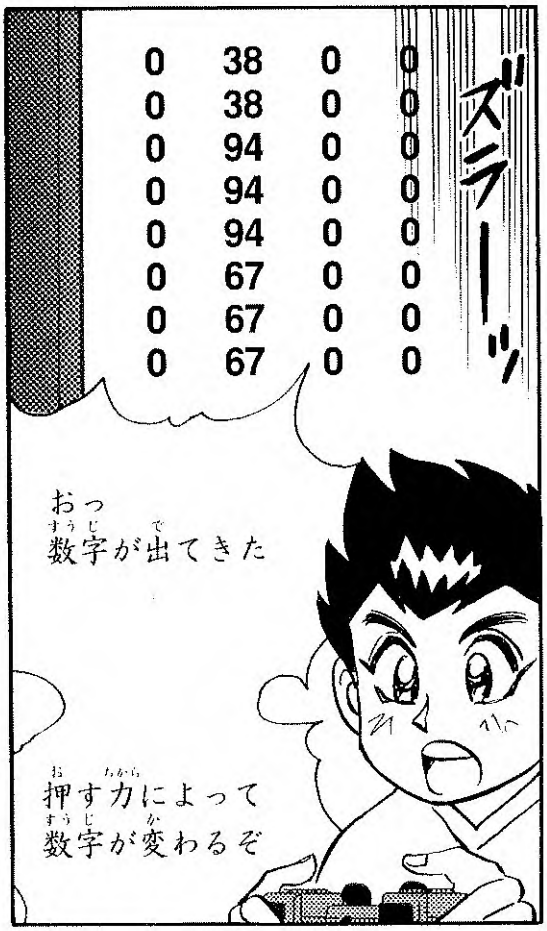
RUN…で
実行と…
ポチッ


実行してごらん
えーと…これは…
コントローラ端子１の
コントローラで押された
ボタンを元に
PAD 関数が返す値を表示
するプログラムかな…

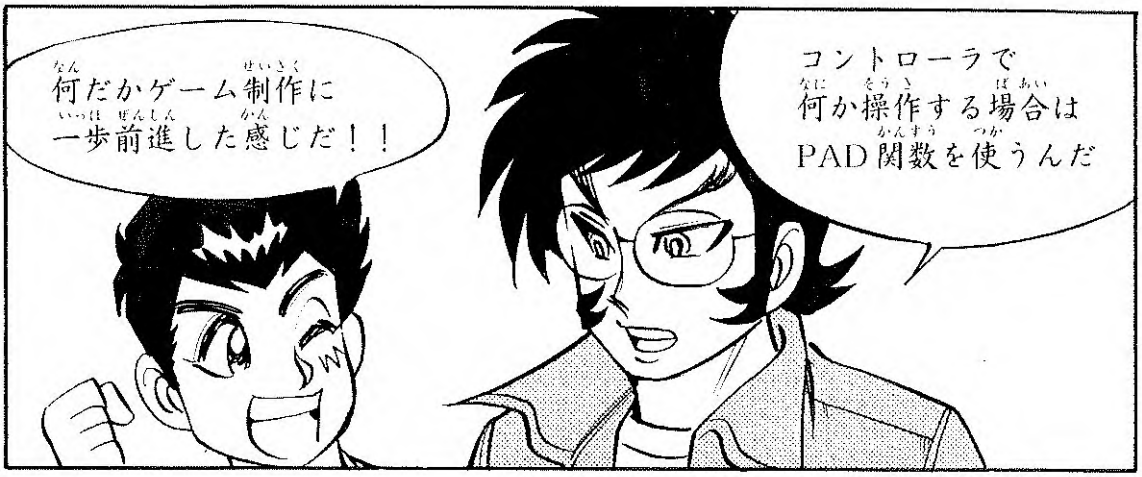
コントローラの
方向キーを押して
みるんだ

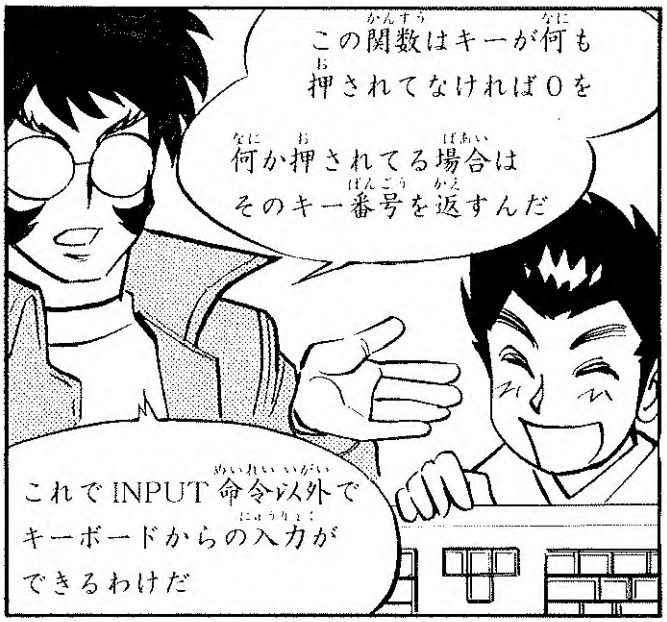
0 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0
実行したら
0が4つ並んで
表示され
続けてるけど…


ほいっ
ポチッ


ズラーッ
0 38 0 0
0 38 0 0
0 94 0 0
0 94 0 0
0 94 0 0
0 67 0 0
0 67 0 0
0 67 0 0
おっ
数字が出てきた
押す力によって
数字が変わるぞ

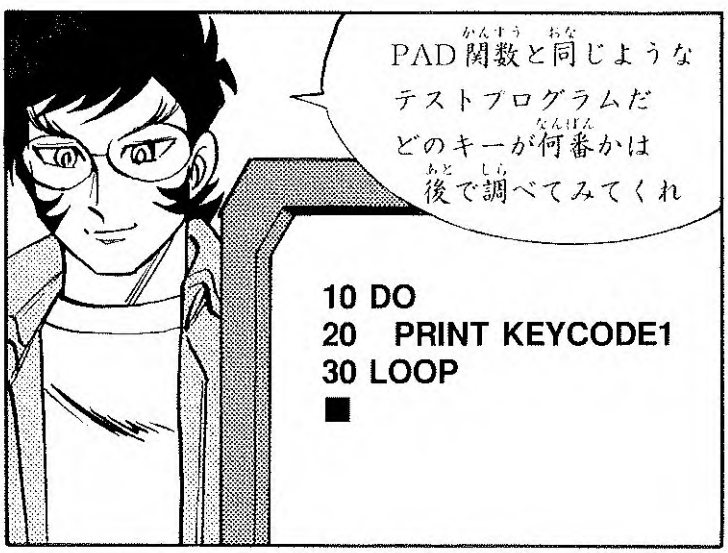
PAD関数は各ボタンの
押され具合に応じて
0から255までの数字が
返って来る
おもしろーい
なるほど！
これでゲームに必要な
コントローラからの
入力ができるのか


何だかゲーム制作に
一歩前進した感じだ！！
コントローラで
何か操作する場合は
PAD関数を使うんだ


この関数はキーが何も
押されてなければ0を
何か押されてる場合は
そのキー番号を返すんだ
これでINPUT命令以外で
キーボードからの入力が
できるわけだ

キーボードの場合
押されているキーを調べる
関数はKEYCODE1だ
引数は無い

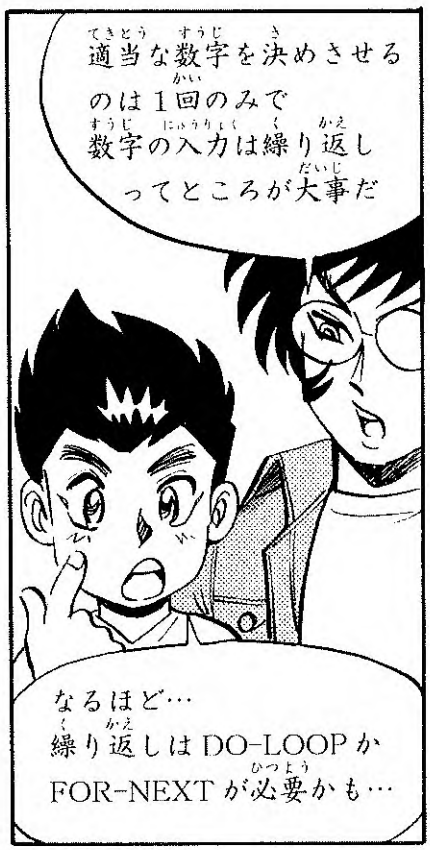
なるほどー
押されたキーで違う
数字が表示されてる
ポチ
ポチッ


PAD関数と同じような
テストプログラムだ
どのキーが何番かは
後で調べてみてくれ
10 DO
20 PRINT KEYCODE1
30 LOOP
■

数字当てゲームを作ろう！

※答えを見る前に挑戦してみよう

1. 適当な数字を考えさせる
2. 当たりだと思う数字を入力する
3. 間違っていたら数字の大小を表示する
4. 正解するまで数字を入力する



```
10 A=RND
20 DO
30  INPUT B
40  IF A 〉B THEN PRINT "もっと大きい"
50  IF A 〈 B THEN PRINT "もっと小さい"
60 LOOP UNTIL A = B
70 PRINT "正解"
```

※日本語の入力方法はUser's Manualをご覧下さい

こんな感じでどうだ！

```
10 A=RND
20 DO
30  IN
40  IF A
```

10行で変数Aに
RND関数の値を代入
してるが
どんな数字が入る？

うむ…
なかなか良いが
ちょっと間違ってる
ところがあるな…

へっ？

あっ！そうか！！
0から1未満の小数点
付きの数字が入るんだ

そうだ
何倍かにして小数点
以下を切り捨てた
方が良い

10 A = INT(RND * 100)
20 DO

これなら0から100未満の整数になるはずだ

とりあえず直して入力だ

タン タン

RUN…と

カチャ カチャ

うむ…良いようだな実行してみろ

50
もっと大きい
?
■

おっ50よりも大きいんだ

とりあえず50…と

カチッ

Enter

RUN
?
■

よし数字を聞いて来た

?
51
正解
■

やった

じゃこれで…

もっと小さい
?
54
もっと小さい
?
■

カタ カタ

もう少し？

70
もっと小さい
?
60
もっと小さい
?
■

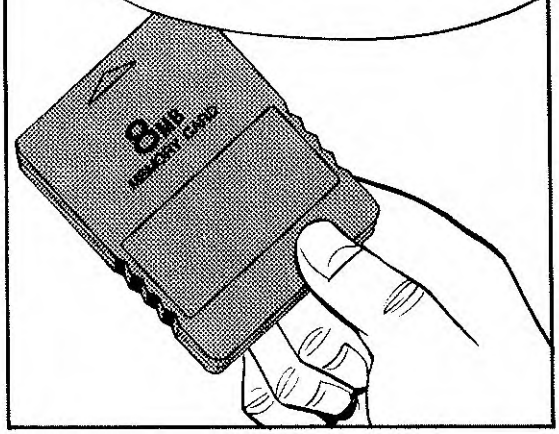
カタ カタ

くそー

?
70
もっと小さい
?
■

良くできたぞ
でも…まだ問題があるんだけどな
エッヘン
ちゃんと動いたぜ～
カンゲキ～！！
へっ？
実はRND関数は使うたびにランダムな数字を返して来るが
プログラムを実行するたびにリセットされてまた同じ順番でランダムな数字を返してしまうんだ
そんなときはRANDOMIZE命令を使ってRND関数が返す数字をバラバラにすることができる
RANDOMIZE命令
つまり何度実行しても同じ答えになるのさ
この場合は51
それじゃゲームじゃないよ

今作ったプログラムを
"プレイステーション2"専用
メモリーカード(8MB)に記録する
方法を教えて今日は終わりに
しよう



プログラムの最初に
1回だけ実行しておけば
RND関数の返す値が
バラバラになるぞ

```
5 RANDOMIZE
10 A = INT(RND * 100)
20 DO
```

なるほど
そんな便利な
命令もあるんだ

反対に今記録した
GAME1を読み込む
命令はLOADだ

```
LOAD "GAME1"
```

これでGAME1という
名前のプログラムを
読み込むんだ

記録にはSAVE
命令を使う

```
SAVE "GAME1"
```

これでGAME1という
名前でプログラムが
記録される

名前は8文字までの
アルファベットと
数字が使える

ハハハハッ

まだまだ
道のりは長いぞ
おさらいを忘れるな

これで
天才プログラマに
また一歩前進だ

記録がSAVE命令で
読み込みがLOAD命令か
覚えたぜ！！

見やすいプログラムを書こう

見やすいプログラム

書いているプログラムの行数が増えてくると、どの処理が何行にあるかを覚えておくのはとても難しいことです。また、変数の使い道を忘れてしまったり、何をする副プログラムか忘れてしまうこともあるかもしれません。ここでは、それらの問題をできる限り減らすために、見やすくわかりやすいプログラムの書き方を紹介します。

書き方を工夫しよう

◆インデント

・行頭に空白を挿入して字下げすることをインデントといいます。インデントを使うと、繰り返す部分や副プログラムの定義部分がよりわかりやすくなります。

例:

インデントがない場合

```
100 DO
110 CLEAR
120 FOR A=1 TO 10
130 IF PAD(1,A)>0 THEN
140 PRINT "BUTTON";A,"ON"
150 END IF
160 NEXT A
170 FSYNC 1
180 LOOP UNTIL PAD(1,20)=1
```

インデントがある場合

```
100 DO
110  CLEAR
120  FOR A=1 TO 10
130   IF PAD(1,A)>0 THEN
140    PRINT "BUTTON";A,"ON"
150   END IF
160  NEXT A
170  FSYNC 1
180 LOOP UNTIL PAD(1,20)=1
```

このプログラムはどちらもまったく同じ内容ですが、インデントがない場合、FOR～NEXT命令が繰り返す行や、DO～LOOPで繰り返す一連の行をひと目で把握することが難しいものとなっています。
一方、インデントを入れるとどの行がFOR～NEXT命令で繰り返すかなど、ひと目で把握しやすくなります。例えば120～160行のFOR～NEXT命令で繰り返す部分はFOR行より1字以上インデントされた部分であることがわかります。

・インデントは、DO～LOOP命令やIF THEN～END IF命令、SELECT CASE～END SELECT命令、副プログラムなど、繰り返しやプログラムのまとまりを扱うものについて活用することができます。

注釈

・注釈には、REM命令と行末注(!命令)があります。
これらの注釈はプログラムの実行時、何もせずに無視されますが、注釈を活用することでより見やすいプログラムを書くことができます。

◆注釈をメモとして活用しよう

・REM命令を使って、プログラムの機能や動作についてコメントを入れると、プログラムがどのような内容なのかわかりやすくなります。
・行末注(!命令)を使って、その行が何を実行するのかコメントを入れると把握しやすくなります。
例:

注釈がない例

```
100 DO
110  IF PAD(1,6)>0 THEN EXIT DO
120 LOOP
```

注釈を入れた例

```
100 REM ○ボタンが押されるまで待つ
110 DO
120  IF PAD(1,6)>0 THEN EXIT DO ! ○ボタンが押されたら繰り返しを出る
130 LOOP
```

コラム2

◆コメントアウト

・ある処理をする行を、行末注（!命令）を使ってコメントにしてしまうことで、その行を実行させないこともできます。
行を削除しないままプログラムを実行させない手段として、活用することができます。

例:

```
100 DO
110  CLEAR
120 !IF PAD(1,7)>0 THEN
130   PRINT "HAPPY"
140 !END IF
150  FSYNC 1
160 LOOP
```

この例では、120行および140行をコメントアウトして実行させなくしています。これにより×ボタンを押さなくても（IF THEN命令を実行しないで）常に130行が実行されるようになります。
×ボタンを押している間は表示させたい場合、行末注を解除するだけで済み、プログラムを再入力する手間が省けます。

書き方を工夫しよう

◆わかりやすい変数名

・変数名を見て、その使い道がすぐ想像できるような名前をつけるとプログラムがよりわかりやすくなります。
例えば A=A+300 とするより、SCORE=SCORE+300 とした方が、その使い道は簡単に想像することができます。

・配列の要素数などに応用することもできます。

例:

```
2*8の大きさを持つ配列を宣言
100 DIM WPN(2,8) 使い道が不明瞭

変数に値を代入してから配列を宣言
100 PLAYER=2
110 WEAPONS=8
120 DIM WPN(PLAYER,WEAPONS) ある程度使い道が想像できるようになります。
```

◆副プログラムを活用しよう

・似たような処理がプログラム中に複数ある場合、副プログラムとしてまとめるとプログラムの量が少なくなり、全体的に見やすくなります。
また、副プログラムとしてまとめておくと副プログラムだけ修正するだけで済み、複数の行を修正する手間が省けます。
・ある機能をまとめて副プログラムにすると扱いやすくなります。例えば初期化や、キャラクターの座標更新など、機能ごとに分類すると、よりプログラムを把握しやすくなります。

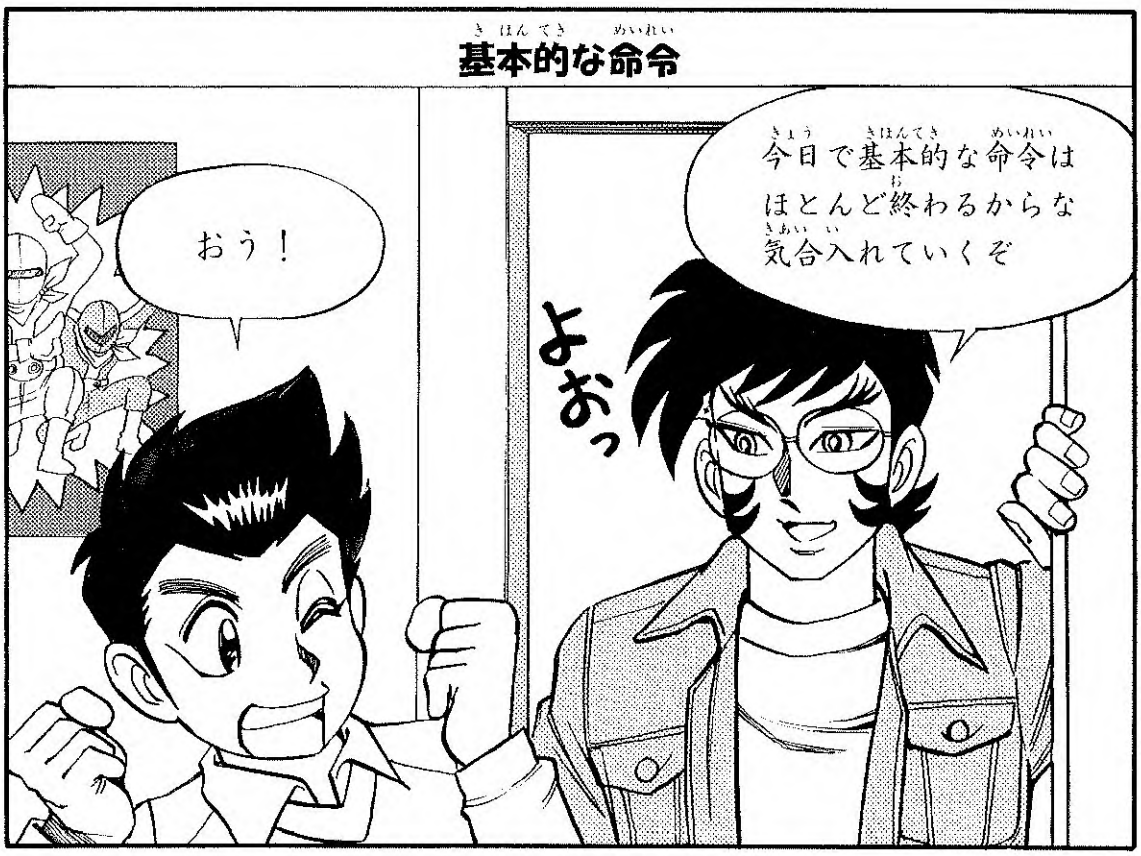
例:

```
100 DO  ! メインループ
110  CALL PAD_CHK(1)  ! コントローラ入力チェック
120  CALL UPDATE_SHIP ! 自機の更新
130  FOR A=1 TO 100
140   CALL UPDATE_ZAKO(A) ! 他キャラクターの更新
150  NEXT A
160 LOOP
...
```

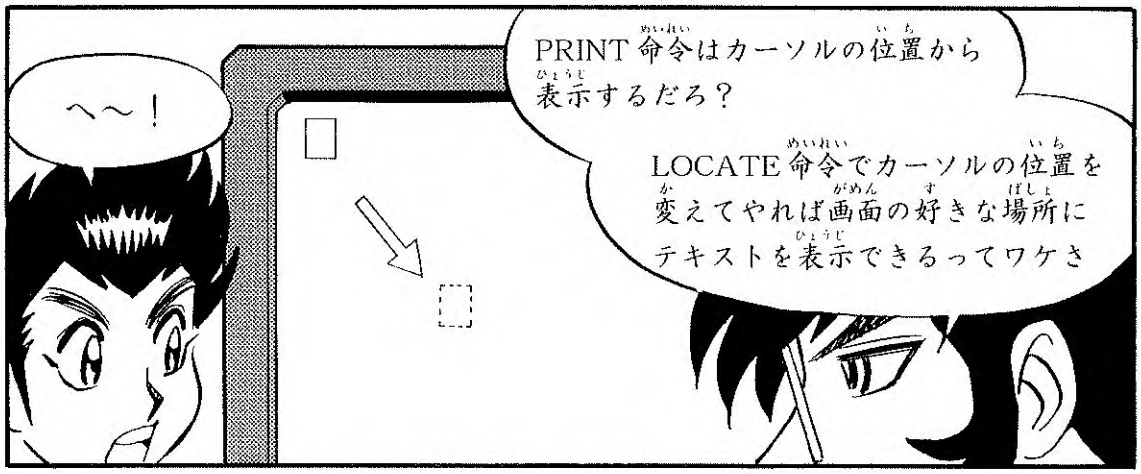
例では、機能ごとに定義された各副プログラムを順番に呼び出して、メインループ（プログラムの中心となる繰り返し）の動作を把握しやすくしています。

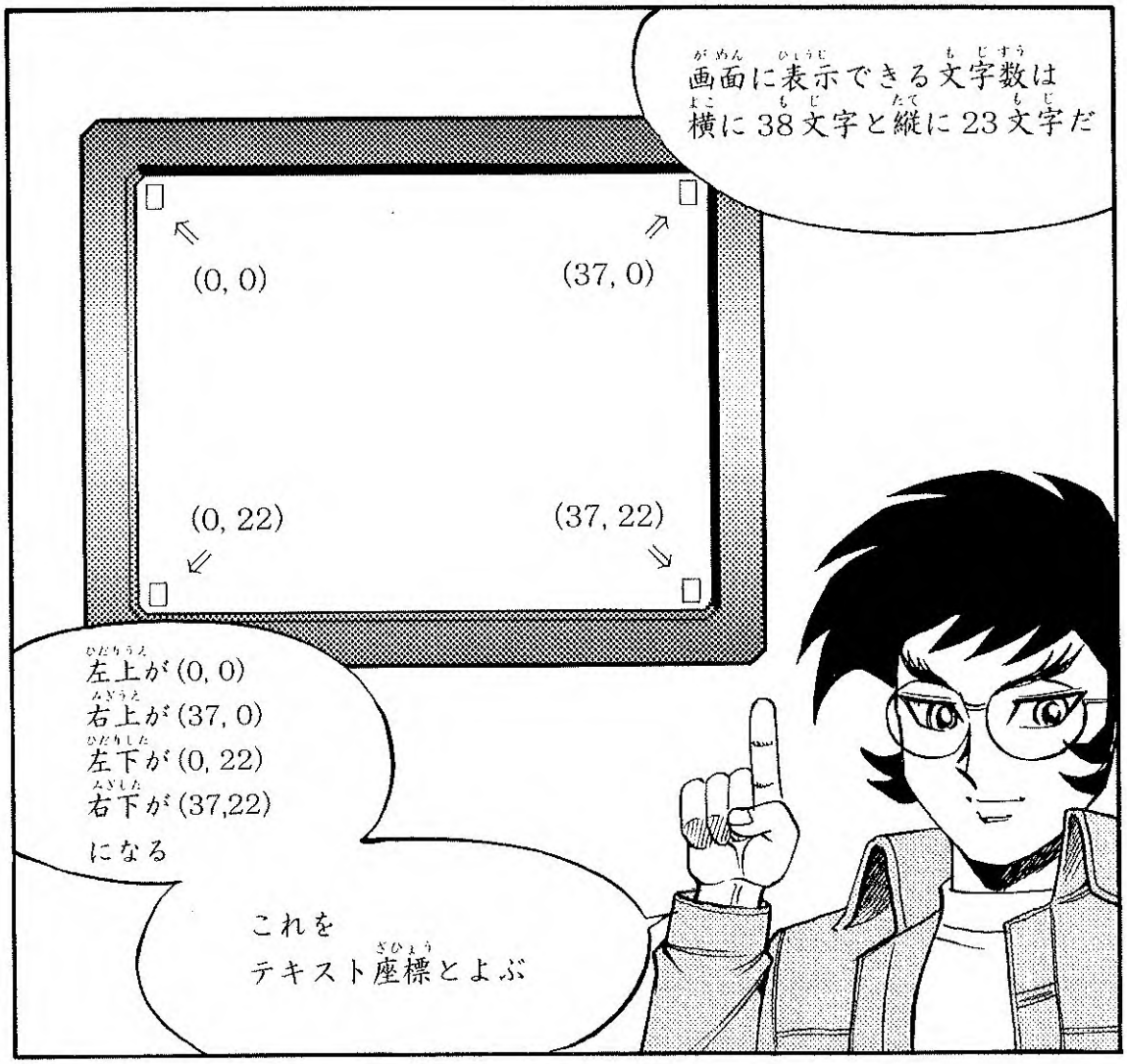
・副プログラム名も、見てわかりやすい名前をつけると把握しやすくなります。
その他、サンプルプログラムやデモプログラムを参考にして、見やすいプログラムを考えてみてください。

基本的な命令


今日で基本的な命令は
ほとんど終わるからな
気合入れていくぞ
よおっ
おう！

まずはLOCATE命令を
教えておこう
これはPRINT命令の前に
使うもので
カーソルの位置を自由に
変更できるんだ


へ～！
PRINT命令はカーソルの位置から
表示するだろ？
LOCATE命令でカーソルの位置を
変えてやれば画面の好きな場所に
テキストを表示できるってワケさ


画面に表示できる文字数は
横に38文字と縦に23文字だ
(0, 0)
(37, 0)
(0, 22)
(37, 22)
左上が(0, 0)
右上が(37, 0)
左下が(0, 22)
右下が(37,22)
になる
これを
テキスト座標とよぶ

横の表示文字数を50に増やす命令もあるが今は置いておく

座標は1からじゃなくて0から数えるからな
0～37で38文字だ
なるほど！！

ん？横が38文字なのに最大が37って…？

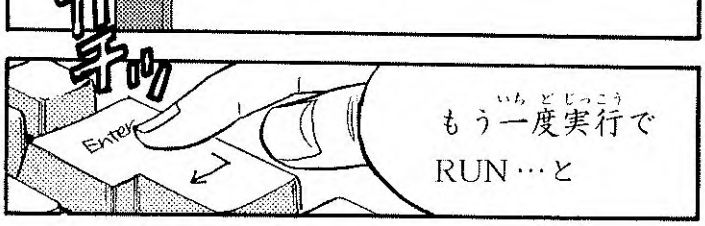
でLOCATE命令の
使い方はこうだ
LOCATE 20,10
これで横20縦10の
位置にカーソルを
移動できる
LOCATE命令の前に
ESCキーで画面を
クリアしておこう
カチッ
ESC
なるほど！
この次にPRINT命令で
テキスト表示させれば画面の
自由な位置に表示できるわけさ
TEST
TEST
(20,10)
むむ…
また問題か…
よ～し
よし、じゃあ
画面の左端から右端まで
テキストが移動するプログラムを
書いてみろ
※答えを見る前に挑戦してみよう

ちょっと書いてみよう
表示するテキストは
TEST でいいかな
10 A=0
20 LOCATE A,0
30 PRINT "TEST"
■
それから…と
えーと LOCATE を使って
カーソルの位置を決めて…
移動するわけだから
変数を使えばいいな…
位置を変数にいれて
LOCATE で変数の位置に
カーソルを移動させよう
変数に 1 を足しながら
繰り返せば位置を変え
ながら表示できるな
DO-LOOP 命令で
繰り返しさせよう
カチャ
カチャ
できたら
実行してみろ
RUN…と
カチャ
15 行に DO を追加で
変数に 1 を足しながら
LOOP で繰り返すと
10 A=0
15 DO
20 LOCATE A,0
30 PRINT "TEST"
40 A=A+1
50 LOOP
■
こんな感じで
どうだ？

```
TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT
EST　を開始します。

20 LOCATE A,0
座標指定の誤りです。
指定位置が画面範囲を超えています。
```

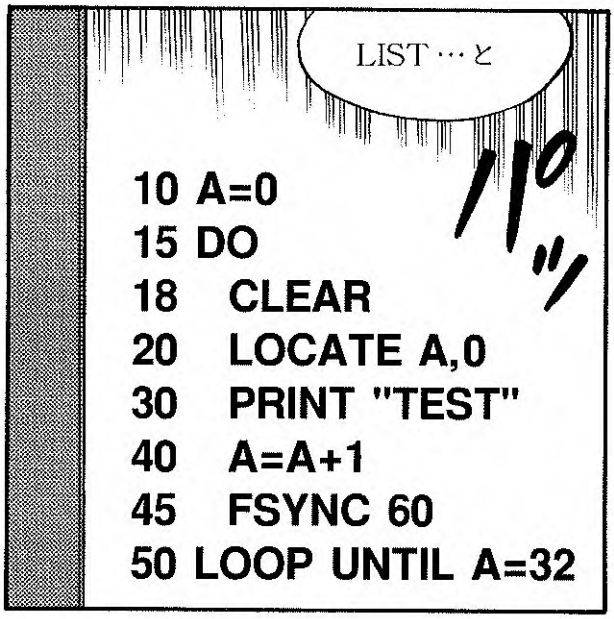
```
10 A=0
15 DO
20   LOCATE A,0
30   PRINT "TEST"
40   A=A+1
45   FSYNC 60
50 LOOP
```

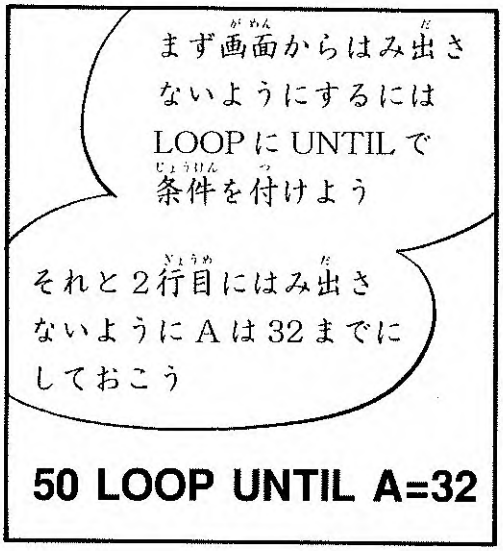


なるほど…
Tが並んでたのは
TESTのTが残っている
からだったのか
TTEST
TTTTTTTEST
おおっ ゆっくり
実行され始めた
TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT
EST を開始します。
20 LOCATE A,0
座標指定の誤りです。
指定位置が画面範囲を超えてい
端っこまで行ったら
エラーになった…
正解だな
じゃあ修正する前に
画面をクリアする
命令を教えておこう
あっそうか！！
LOCATEで使ってる
変数Aが
画面横サイズの37
を超えたんだな

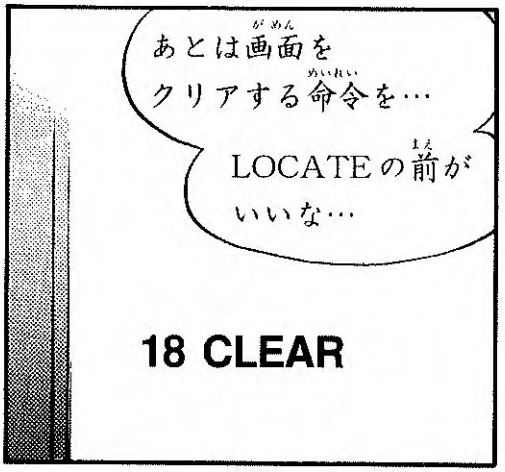
了解！
余計なTを
消すのにそいつを
使って修正しよう
CLEAR命令は
実行するとカーソルの
位置が左上(0,0)になって
しまうから注意しろよ

CLEAR命令で
画面のテキストは
すべてきれいに消えるぞ
CLEAR


LIST…と
10 A=0
15 DO
18 CLEAR
20 LOCATE A,0
30 PRINT "TEST"
40 A=A+1
45 FSYNC 60
50 LOOP UNTIL A=32


まず画面からはみ出さ
ないようにするには
LOOPにUNTILで
条件を付けよう
それと2行目にはみ出さ
ないようにAは32までに
しておこう
50 LOOP UNTIL A=32

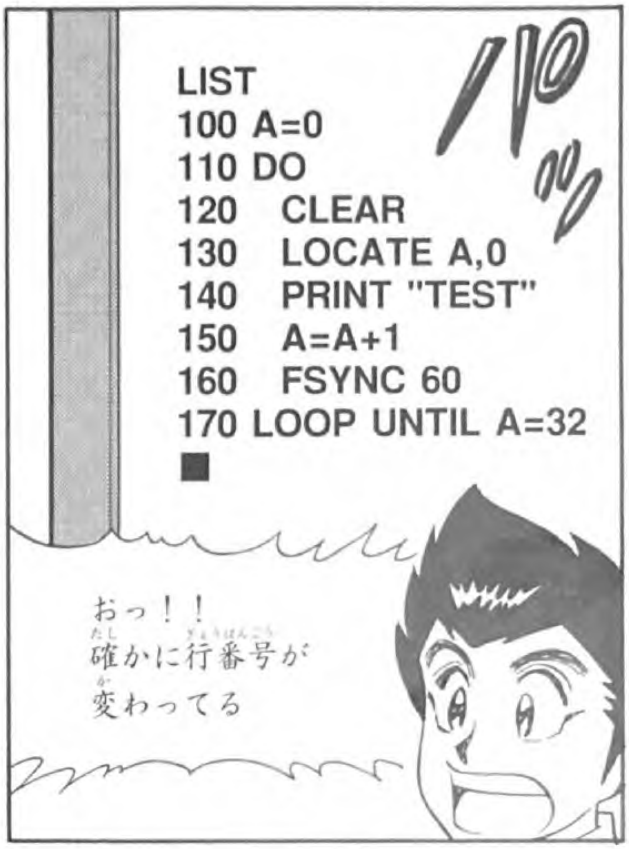
今度は大丈夫
かな…
RUN…と
カチャカチャ

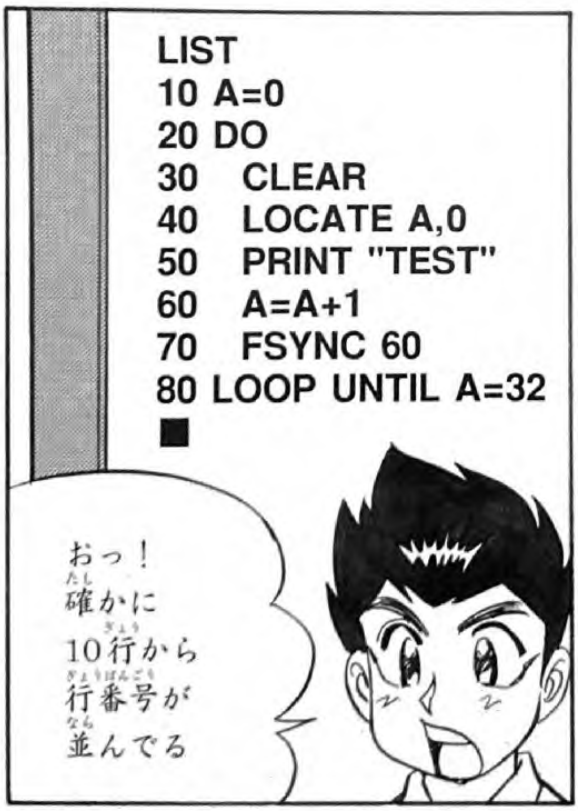

あとは画面を
クリアする命令を…
LOCATEの前が
いいな…
18 CLEAR

うむ
よくできたな
おっし！！
うまく行ったぜ！
TEST
TEST
TEST
良いことを教えてやろう
行番号を並び替える
命令もあるんだ
RENUMBERって
書いて実行してみろ
でもなんだか
行を追加したら
行番号の並びが
汚くなったな
10 A=0
15 DO
18 CL
20 LO
30
LIST命令で
見てみよう
タンタン
RENUMBER
OK
■
これでいいの？

でも…
なんで100行から
始まってるの？

始めの行番号を
変更したいときには
RENUMBERの後に
ATと書いて番号を
書くんだ


LIST
100 A=0
110 DO
120 CLEAR
130 LOCATE A,0
140 PRINT "TEST"
150 A=A+1
160 FSYNC 60
170 LOOP UNTIL A=32
■
おっ！！
確かに行番号が
変わってる


LIST
10 A=0
20 DO
30 CLEAR
40 LOCATE A,0
50 PRINT "TEST"
60 A=A+1
70 FSYNC 60
80 LOOP UNTIL A=32
■
おっ！
確かに
10行から
行番号が
並んでる

こんな感じだな…
RENUMBER AT 10
OK
■
LIST命令で
表示してみよう
ﾀﾗｰﾝ

ゲームでは敵の移動や
自分の移動などが
よく使われるからな
そういった部分は
今のプログラムを
応用するんだ

KEYCODE1やPAD命令と
組み合わせれば
自由に動かせそうだな

配列変数
じゃあ次は
配列について
教えておこう
?
配列？

いいところに
目を付けたな
そのとおりだ
このプログラムは
ここまでにしよう

変数を数字を入れる
箱と考えると

配列ってのは
その箱がたくさん
連なってるようなものだ

← 変数

← 配列

配列ってのは
配列変数のことだ

普通の変数は1つの数字や
文字列しか代入できない

ふんふん

変数を使う場合は特に命令が
必要なわけではないが

配列を使う場合は
DIM命令で配列変数を
用意しなければならない

後ろの括弧にある10は
10個の箱を用意しろと
いうことだ

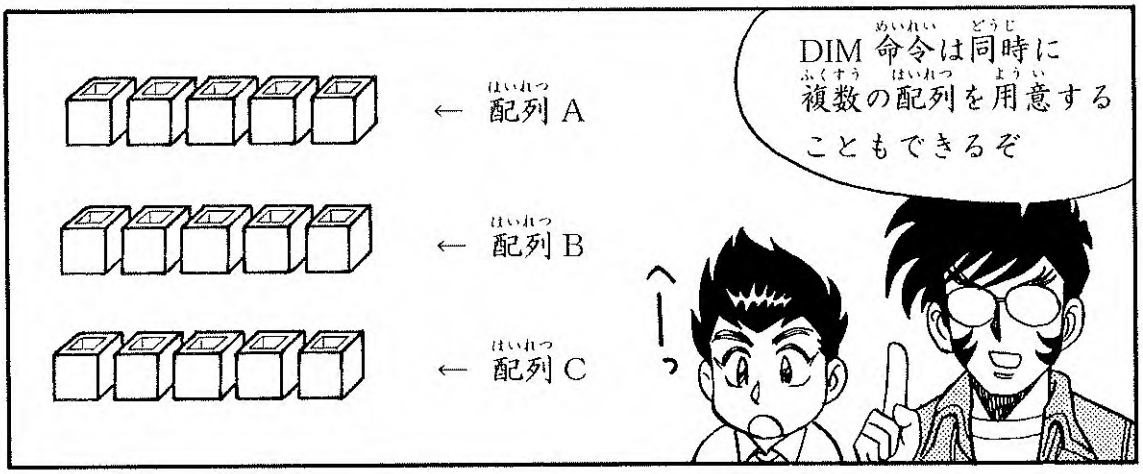
A(1) …… A(5) …… A(10)

これでA(1)～A(10)の
10個の変数が使えるんだ

こんな感じだ

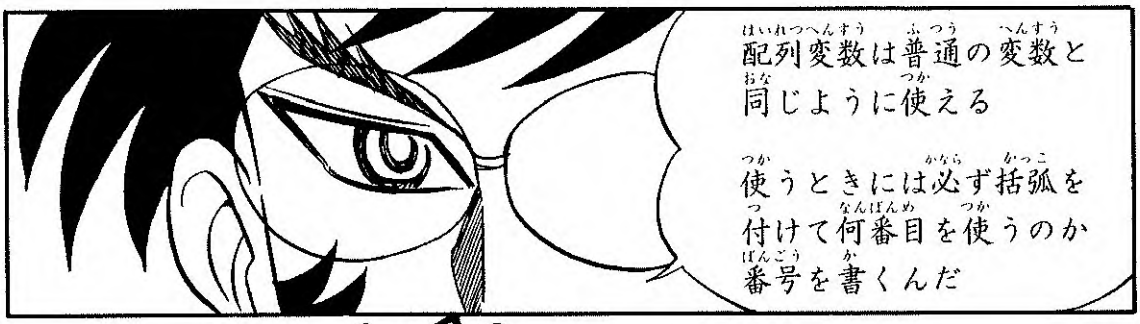
DIM A(10)

これでAという
配列変数が用意された
ことになる



こう書くんだ

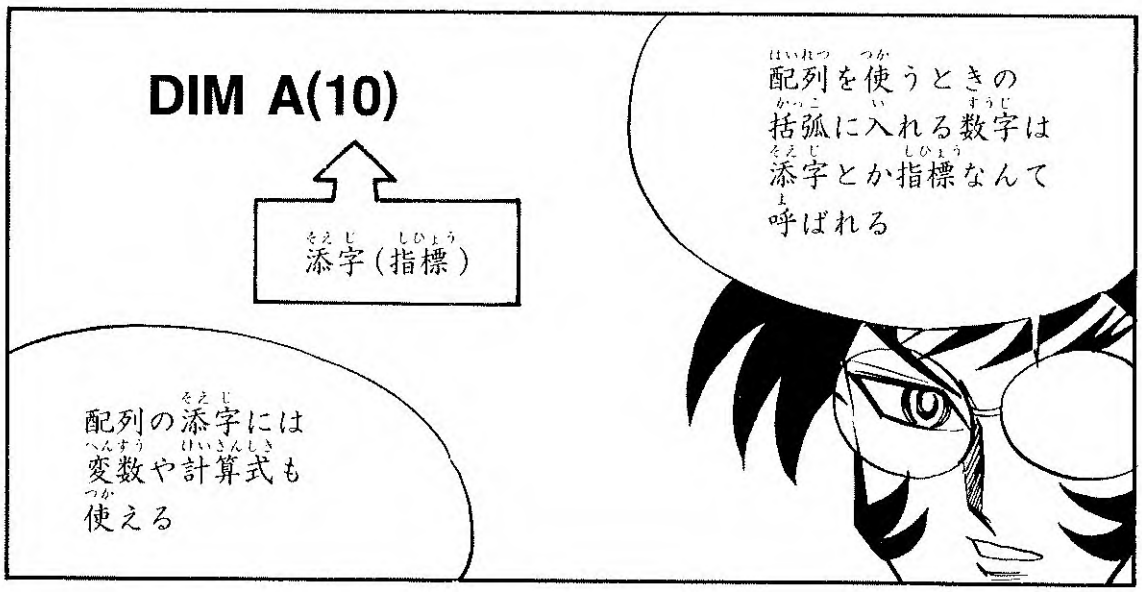
```
DIM A(5), B(5), C(5)
```

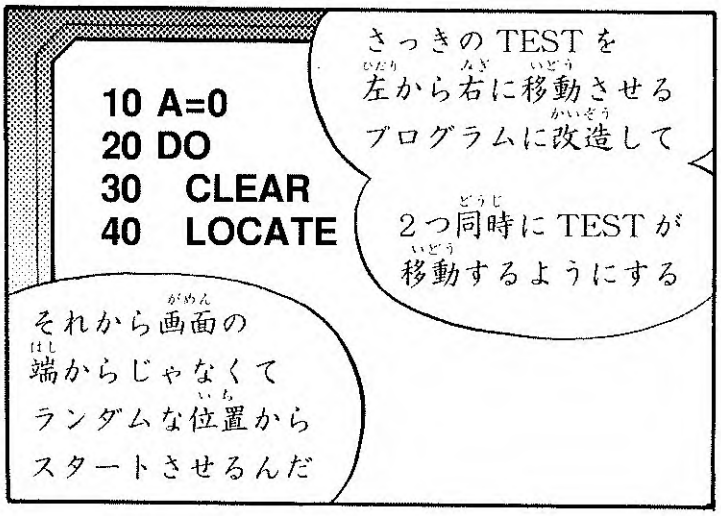


こんな感じだ

```
10 DIM A(10)
20 A(1)=10
30 A(2)=20
40 PRINT A(1)+A(2)
■
```

カタカタ





※答えを見る前に挑戦してみよう

さっきのプログラムは
これだな

```
LIST
10 A=0
20 DO
30   CLEAR
40   LOCATE A,0
50   PRINT "TEST"
60   A=A+1
70   FSYNC 60
80 LOOP UNTIL A=32
■
```

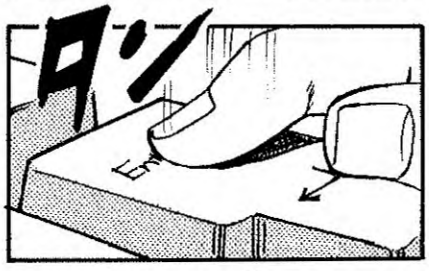
```
100 A=INT(RND * 10)
110 B=INT(RND * 10)
■
```

変数を2つ用意して
スタート位置は
ランダムに…

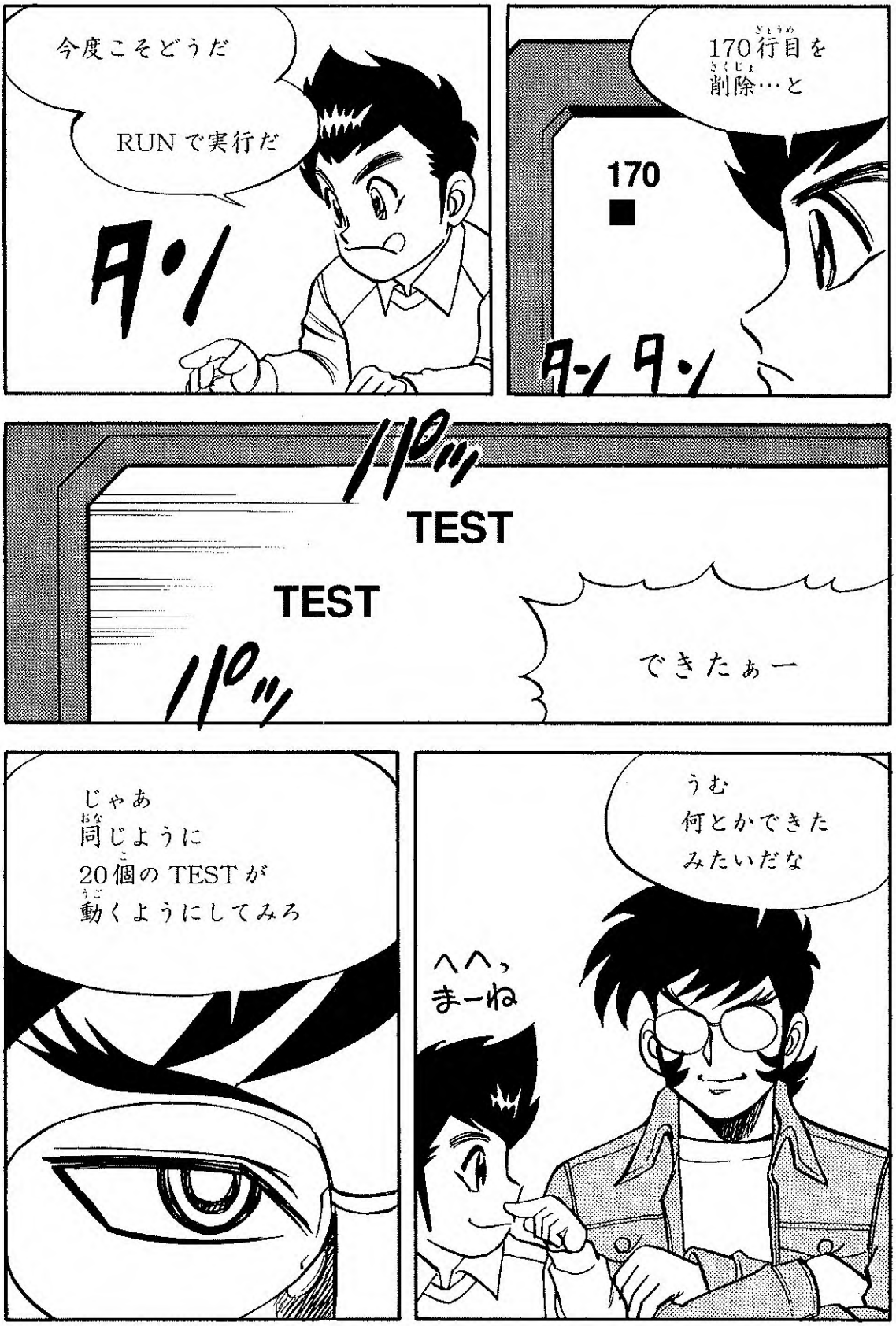
```
210   FSYNC 60
220 LOOP UNTIL A=32 OR B=32
```

LOOPの終了条件は
どうしよう…
変数のどちらかが32に
なったら終わりにしよう

カタ
カタ
カタ

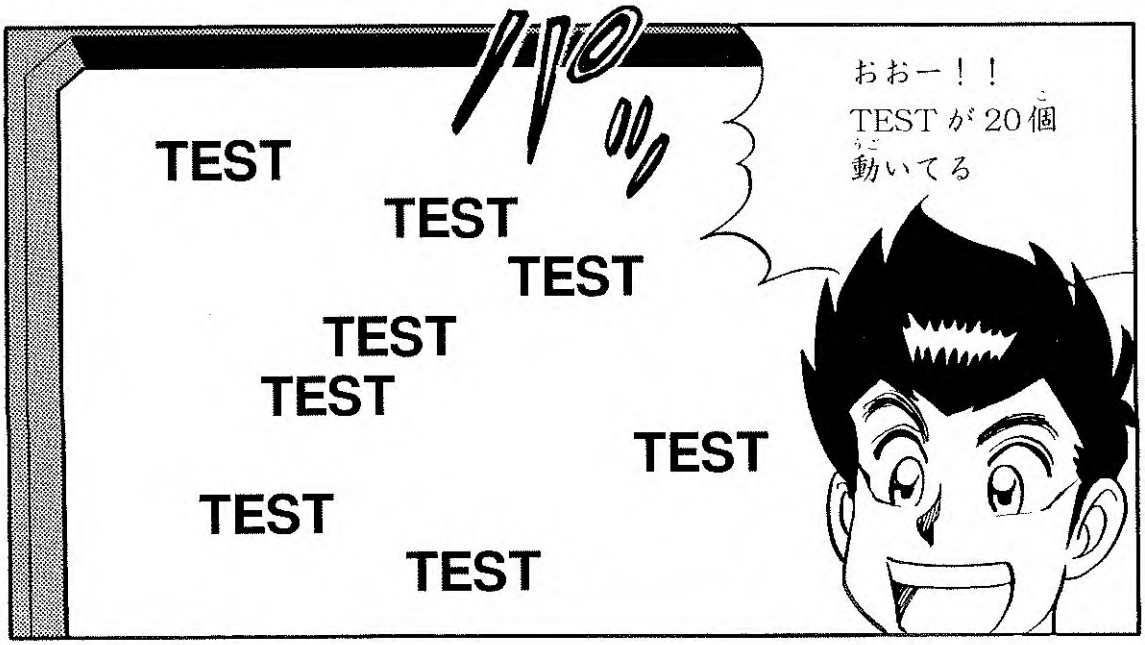



```
LIST
100 A=INT(RND * 10)
110 B=INT(RND * 10)
120 DO
130   CLEAR
140   LOCATE A,0
150   PRINT "TEST"
160   A=A+1
170   CLEAR
180   LOCATE B,1
190   PRINT "TEST"
200   B=B+1
210   FSYNC 60
220 LOOP UNTIL A=32 OR B=32
```

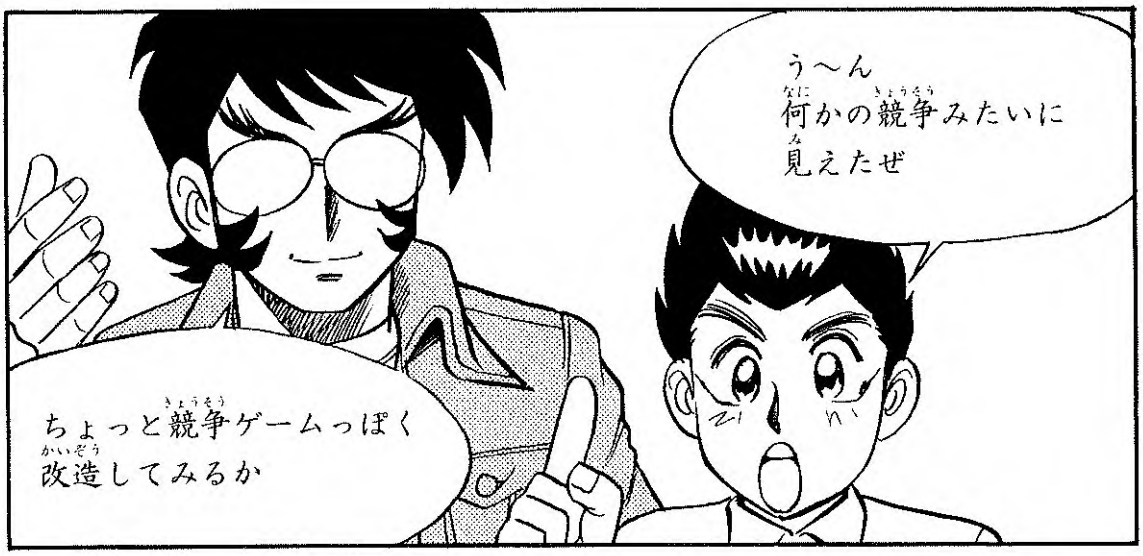

今度こそどうだ
RUN で実行だ
タン
170行目を
削除…と
170
タン タン
TEST
TEST
できたぁー
じゃあ
同じように
20個の TEST が
動くようにしてみろ
うむ
何とかできた
みたいだな
へへっ
まーね

えーと
変数を20個用意して…
プログラムもかなり
長くなるなぁ…
げげっ！！
そんなにたくさん
動かすのか…
うお～っ！！
20個も同じような
プログラムを書くのが
面倒だぁ～
試しに
書いてやろう
これを実行して
みるんだ
ﾀﾀﾝ
そこでだ！
さっき教えた配列変数で
便利になるんだ

```
100 DIM X(20)
110 FOR Y=1 TO 20
120   X(Y)=INT(RND * 10)
130 NEXT Y
140 F=0
150 DO
160   CLEAR
170   FOR Y=1 TO 20
180     LOCATE X(Y),Y
190     PRINT "TEST"
200     X(Y)=X(Y)+1
210     IF X(Y)=32 THEN F=1
220   NEXT Y
230   FSYNC 60
240 LOOP UNTIL F=1
■
```



2箇所FOR-NEXT命令があるだろう
変数Yを配列の添字にして20個の配列変数を一度に処理してるからな
どれか一つが画面の端に付いたら変数Fに1を入れてループを終了させているんだ
110 FOR Y=
120 X(Y)=IN
130 NEXT Y
170 FOR Y=1
180 LOCAT
190 PRINT
プログラムもぜんぜん長くなってないな…
配列変数を使えば100個動かしたってプログラムは長くならないからな
ところでさっきの画面を見て何か思わなかったかい？
TEST
TEST
TEST
TEST
うん！確かに便利だ
テキストは縦に100個も表示できないからこの例では無理だけど
これで配列変数の便利さがわかっただろう



```
100 DIM X(20)
110 RANDOMIZE
120 F=0
130 DO
140   CLEAR
150   FOR Y=1 TO 20
160     LOCATE X(Y),Y
170     PRINT "O"
180     IF INT(RND * 3) > 1
 THEN X(Y)=X(Y)+1(←180行の続き)
190     IF X(Y)=32 THEN F=1
200   NEXT Y
210   FSYNC 60
220 LOOP UNTIL F=1
■
```

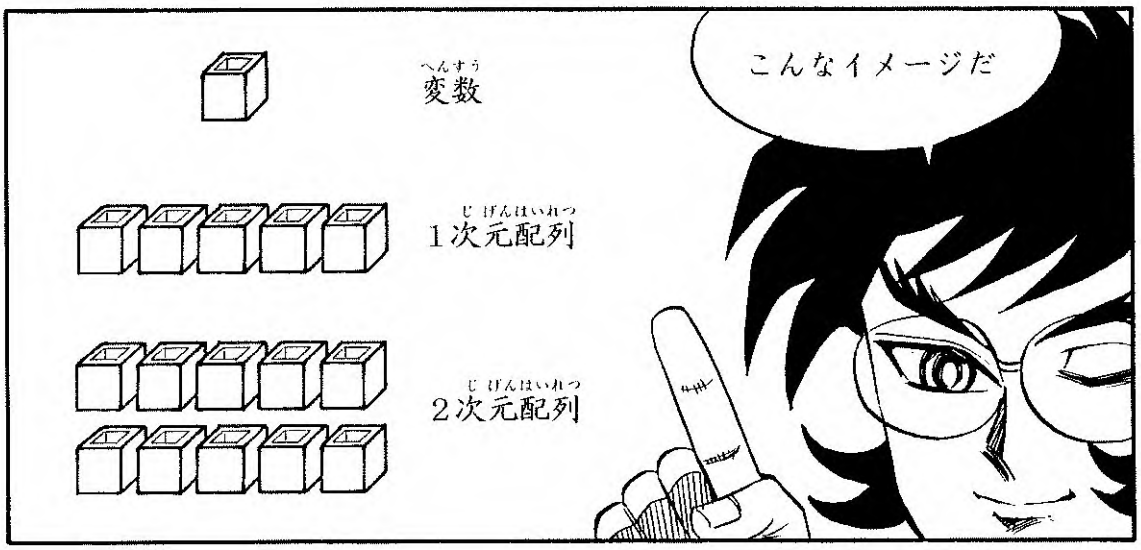
おーっ
競争ゲームっぽく
なったぜー

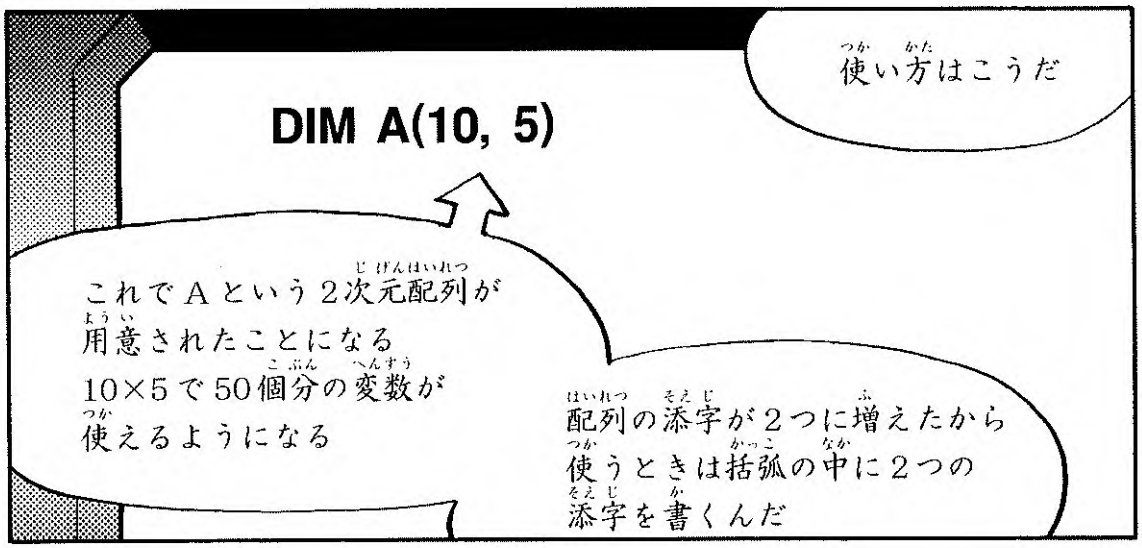
これを少し改造すれば
競馬ゲームとかも
作れそうだぜ

今回は配列変数を
教えるための例として
作っただけだから
ここまでにしておこう

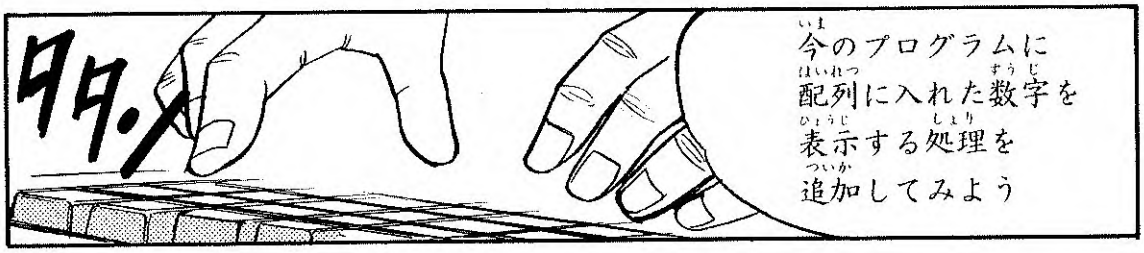
まあ色々と改造して
行けば作れるぞ
基本部分はこんな
感じだからな

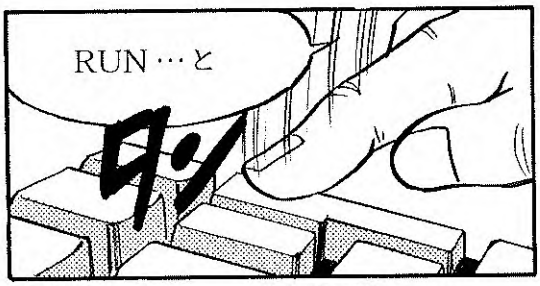
配列には添字が１つで
ないものもある
添字が２つあるものは
２次元配列と呼ぶ
それに対して添字が１つの
配列は１次元配列というんだ


こんなイメージだ
変数
１次元配列
２次元配列


使い方はこうだ
DIM A(10, 5)
これでＡという２次元配列が
用意されたことになる
10×5で50個分の変数が
使えるようになる
配列の添字が２つに増えたから
使うときは括弧の中に２つの
添字を書くんだ

おや？
FOR-NEXT命令が
2重になってる
そうだな
ループを2重にして
XとYの2つの変数を
添字にすることで
全ての配列変数に
数字を入れてるんだ
100 DIM A(10, 5)
110 FOR Y = 1 TO 5
120 FOR X = 1 TO 10
130 A(X, Y) = INT(RND * 5)
140 NEXT X
150 NEXT Y
これは10×5の
2次元配列に
ランダムな数字を
入れるプログラムだ
色々な使い方があるぞ
例えばゲームでは
何らかのマップを表示する
ことが多いだろう？
なるほど…で
2次元配列ってのは
どんなときに使えば
いいの？
RPGとか
パズルとか…
その他にも
色々あるけど
そんなときに
2次元配列を
使うんだ

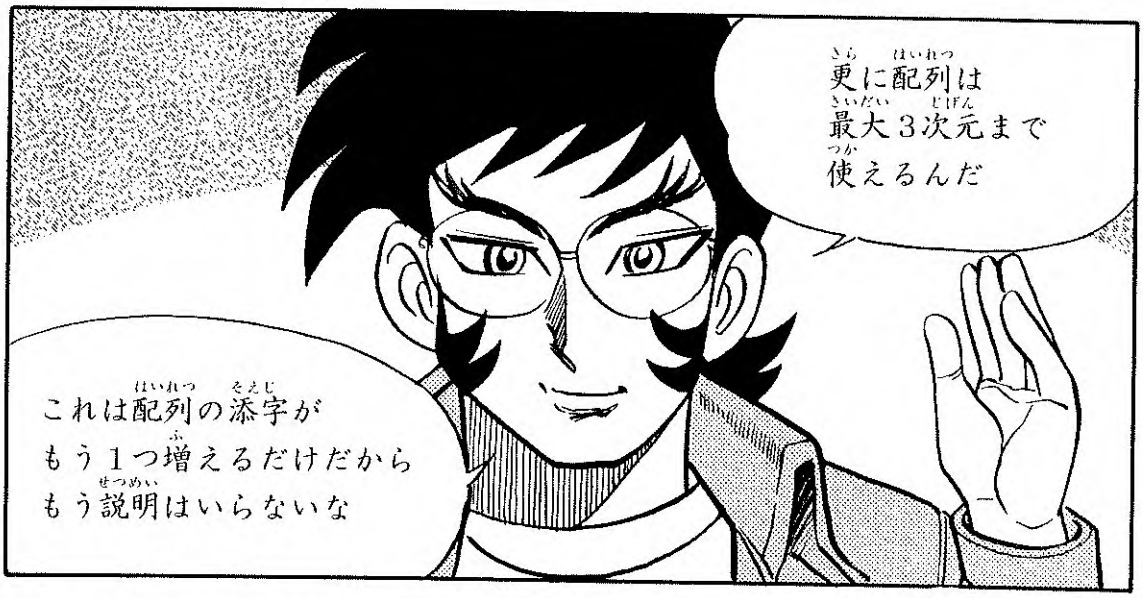





```
100 DIM A(10, 5)
110 FOR Y = 1 TO 5
120   FOR X = 1 TO 10
130     A(X, Y) = INT(RND * 5)
140   NEXT X
150 NEXT Y
160 CLEAR
170 FOR Y = 1 TO 5
180   FOR X = 1 TO 10
190     LOCATE X, Y
200     PRINT A(X, Y)
210   NEXT X
220 NEXT Y
■
```

```
2012403120
0111423323
4031142332
4423341341
1244230414
```

なるほど
でもこれを
グラフィックの木や地面に
変えればRPGのマップっぽく
なるだろう？

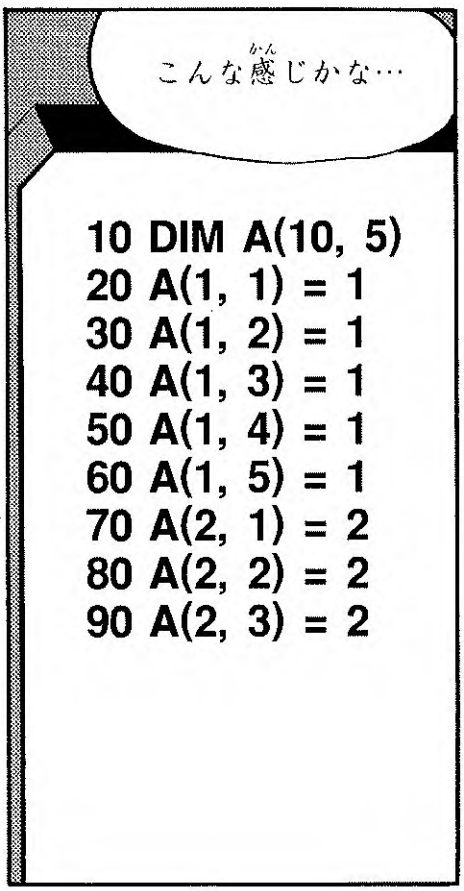

更に配列は
最大3次元まで
使えるんだ
これは配列の添字が
もう1つ増えるだけだから
もう説明はいらないな

配列も変数と同じように
プログラムでは必須だからな
自由自在に使えるようになれよ
了解！！

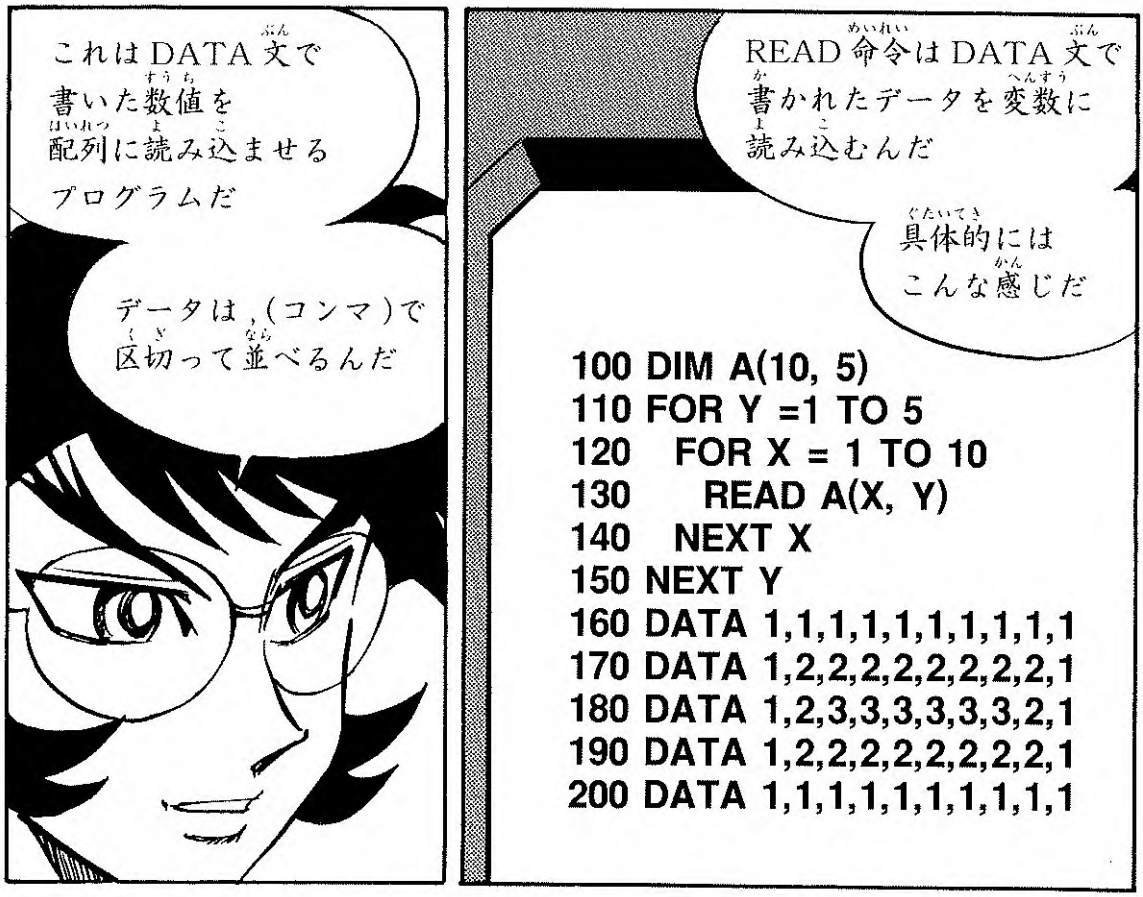
さっきのプログラムは
ランダムな数字マップを
表示したプログラムだが
自分でマップを作る場合は
どうする？

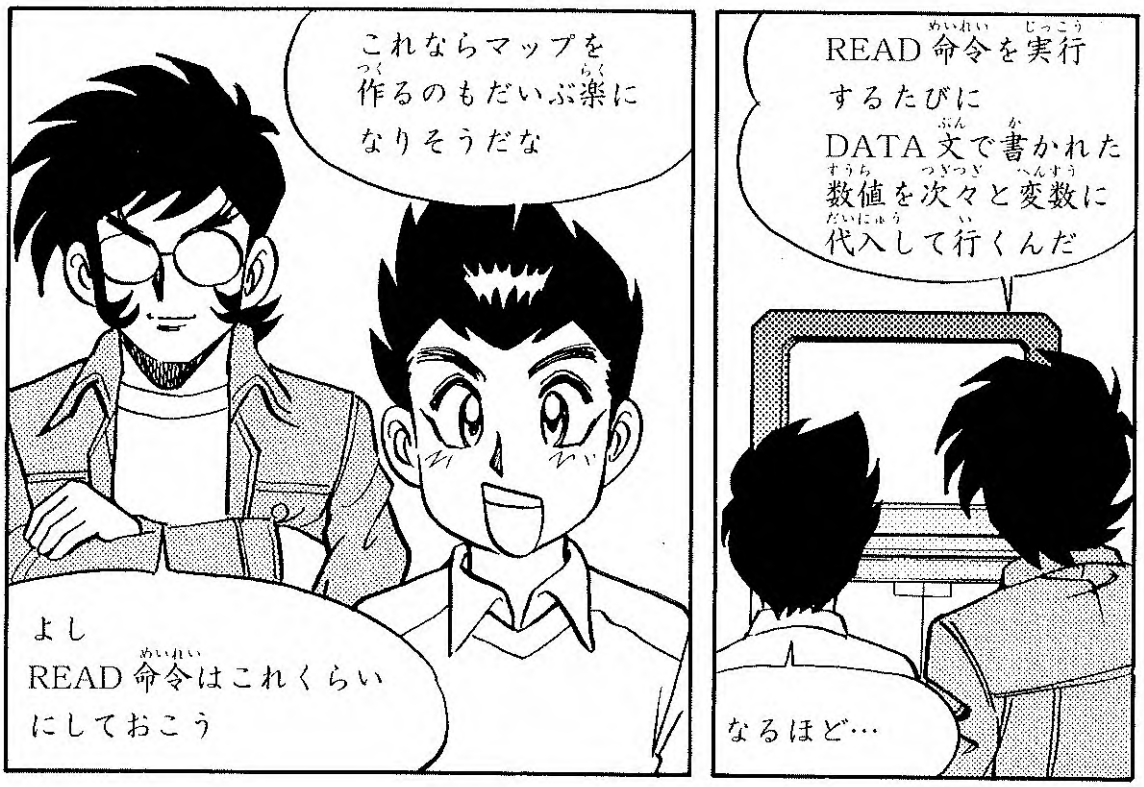
ん～？
配列変数に
値を代入していけば
できそうだけど…

う～ん
でもこれで
マップを作るのは
大変だなぁ…


こんな感じかな…
10 DIM A(10, 5)
20 A(1, 1) = 1
30 A(1, 2) = 1
40 A(1, 3) = 1
50 A(1, 4) = 1
60 A(1, 5) = 1
70 A(2, 1) = 2
80 A(2, 2) = 2
90 A(2, 3) = 2

そうだな…
こういった場合に
便利なのが
READ命令と
DATA文だ

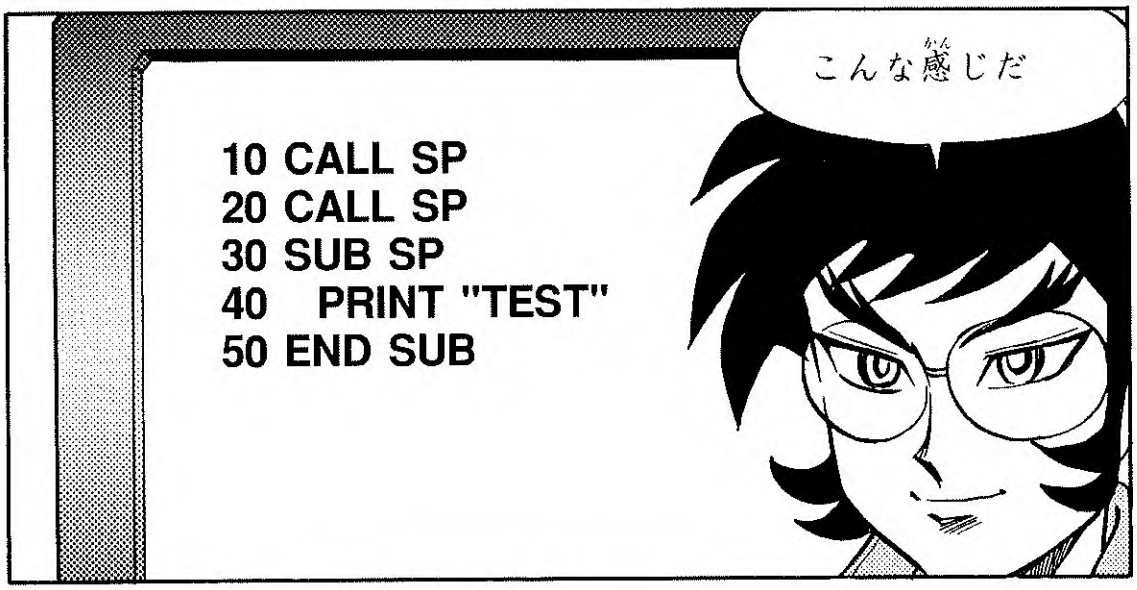

これはDATA文で
書いた数値を
配列に読み込ませる
プログラムだ
データは,(コンマ)で
区切って並べるんだ
READ命令はDATA文で
書かれたデータを変数に
読み込むんだ
具体的には
こんな感じだ
100 DIM A(10, 5)
110 FOR Y =1 TO 5
120 FOR X = 1 TO 10
130 READ A(X, Y)
140 NEXT X
150 NEXT Y
160 DATA 1,1,1,1,1,1,1,1,1,1
170 DATA 1,2,2,2,2,2,2,2,2,1
180 DATA 1,2,3,3,3,3,3,3,2,1
190 DATA 1,2,2,2,2,2,2,2,2,1
200 DATA 1,1,1,1,1,1,1,1,1,1


これならマップを
作るのもだいぶ楽に
なりそうだな
よし
READ命令はこれくらい
にしておこう
READ命令を実行
するたびに
DATA文で書かれた
数値を次々と変数に
代入して行くんだ
なるほど…

ゲームを作っていると
同じようなプログラムを
何度も必要とする
場合が出てくる
そんな時に副プログラム
として定義しておけば
同じプログラムを何度でも
使いまわせるんだ
ふ〜ん

副プログラム
次は
副プログラムに
ついてだ
副プログラム

副プログラムは
SUB命令を使って
定義する
定義された副プログラムは
CALL命令で呼び出す
カタ
カタ


こんな感じだ
10 CALL SP
20 CALL SP
30 SUB SP
40 PRINT "TEST"
50 END SUB

```
10 CALL SP
20 CALL SP
```

この副プログラムは
10行と20行の
CALL命令で
呼び出されてるわけだ

これは30行のSUB命令で
SPという名前を付けて
50行のEND SUBまでの
行を副プログラムとして
定義している

```
30 SUB SP
40   PRINT "TEST"
50 END SUB
```

CALL命令で副プログラムを呼び出すと
SUBの行からEND SUBの行まで実行して
呼び出したCALLの行まで戻ってくるんだ

まあ中身はただ
TESTと表示している
だけだな

つまりこの例では
2回TESTと表示する
わけだな…

プログラム実行中に
CALL命令で呼び出さずに
SUBに来たら
END SUBまで飛び越えるから

CALL命令で呼ばない限り
SUBの中で定義している行は
実行されないんだ

へーなるほど

ふむふむ
でも2回の
CALL命令の後で
30行に来て
SUBが実行されて
3回目のTESTが
表示されたりしない
のかな？

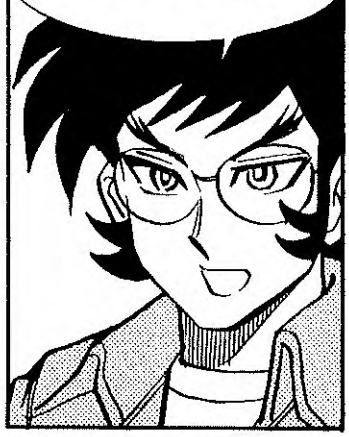
SUB命令で副プログラムの名前を書いた後に括弧で引数を受け取る変数名を書けば良い

```
SUB SP(A)
```

```
CALL(100)
```

CALLの際には副プログラム名の後に括弧で渡す引数を書けば良いんだ

副プログラムには関数のように引数を持たせることもできるんだ



具体的にはこんな感じだな

```
10 CALL SP(100)
20 CALL SP(200)
30 SUB SP(A)
40   PRINT "TEST", A
50 END SUB
```

ふんふんなるほど

この例では引数を変数Aに受け取ってそれをTESTの後に表示してるわけだ

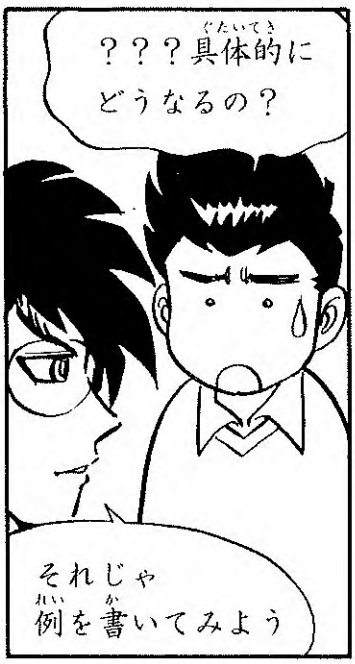
CALL命令の時には引数として100と200を渡している

SUB命令で引数受け取りに使われる変数は局所変数といって
定義したSUBの中でのみ有効なんだ
んん～？？意味がわからないぜ

このときSUB命令で引数の受け取りの為に書かれた変数は今までの変数とはちょっとちがう
へっ？何が違うの？

つまりEND SUBまで実行すると局所変数は破棄されるんだ

さっきのプログラムに25行を足して変数Aを表示させたものだ
この結果がどうなるか予測してごらん
10 CALL SP(100)
20 CALL SP(200)
25 PRINT A
30 SUB SP(A)
40 PRINT "TEST",A
50 END SUB
※答えを見る前に挑戦してみよう


？？？具体的にどうなるの？
それじゃ例を書いてみよう

ん…と
まず10行で引数100を
渡してSPを呼び出す…と

10 CALL SP(100)

30 SUB SP(A)

40 PRINT "TEST", A

SUBで変数Aに
引数100を受け取って
TESTの後ろに
表示させるから
「TEST 100」で…

END SUBでCALL命令に
もどると

50 END SUB

20 CALL SP(200)

20行でも同じように
引数200を渡して
TESTの後ろに表示で
「TEST 200」

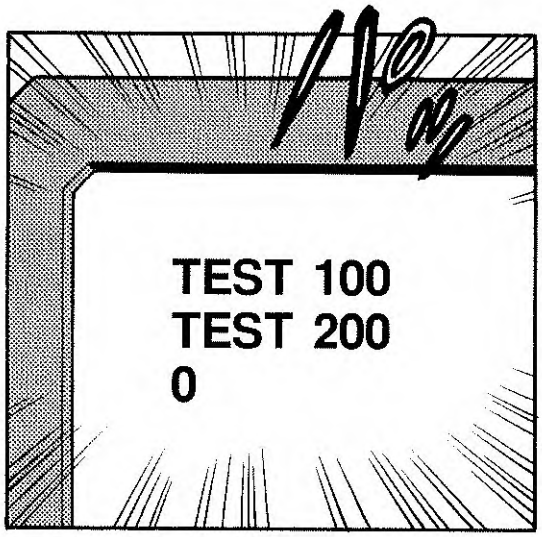
25行で変数Aを表示して
変数Aはさっき渡した引数の
200が入っているハズだから
200を表示…かな…？

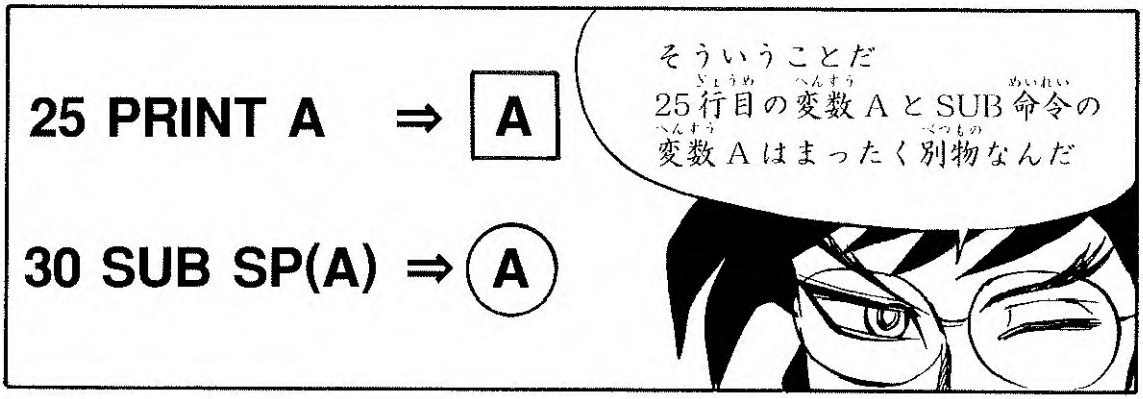
25 PRINT A

というわけで
こんな感じに
表示されるのかな？

TEST 100
TEST 200
200

ん…？
25行で表示している
変数Ａは０なのか？


TEST 100
TEST 200
0


25 PRINT A ⇒ A
30 SUB SP(A) ⇒ A
そういうことだ
25行目の変数ＡとSUB命令の
変数Ａはまったく別物なんだ

25行で表示している
変数Ａは広域変数のＡで
ここで初めて使われた変数
だから中身は０だったんだ
ん～複雑で
よくわからん…

さっき説明した局所変数
に対して今まで使っていた
変数は広域変数とも
言うんだ
きょくしょへんすう
局所変数
こういきへんすう
広域変数

同じ名前の変数を
局所変数として使った場合
SUBの中にいるときは
局所変数の方が使われる
SUBから出たら前に言った通り
局所変数は破棄されて
存在しなくなるから
広域変数を使ってることになる

だいたい
同じ名前の変数が
2つあったら
どっちを使ってるのか
わからないじゃんか？

色々と応用的な使い方が
できて便利なんだが
さらに難しくなるから
このくらいにするか
ホッ

んん～
フクザツで
ムズカシイぜぇ～
この副プログラム関連は
ちょっと難しい所
だからな…

ただもうひとつだけ
命令をおしえておこう

EXIT DOとか
EXIT FORみたいに
SUBから抜ける命令かな…？
正解だ
EXIT SUB命令だ
なんとなく名前から
どんな命令か想像つく
だろう？
EXIT SUBは副プログラムの
途中で脱出して
CALLされた行に戻る事が
できるんだ
簡単な命令だから
説明はいいな
このくらいなら
大丈夫だぜ
まあ　これで
ほぼ基本部分は終了だな
しっかり復習
しておけよ
うい～す！！
了解

コラム3 副プログラムと参照

副プログラムの引数として、数値や文字列に加え各種変数を渡すことができます。
数値や文字列といった値を引数に渡す場合と各種変数を引数に渡す場合では、扱いが若干異なります。
ここでは引数に値を渡す場合と、変数を渡す場合の処理の違いについて説明します。

値渡しと参照渡し

◆値渡し

・副プログラムに数値または文字列といった値を渡します。これを「値を渡す」といいます。
・渡された値は、SUB行で宣言した副プログラムの引数に代入されます。この引数を仮引数と呼びます。
仮引数は副プログラム内部でのみ有効な変数（局所変数）で、その副プログラムの実行が終了するとともに破棄されます。

例:

```
100 X=10
110 CALL ADD(10)
120 PRINT X
130 SUB ADD(X)
140  X=X+5
150 END SUB

＜実行結果＞
10
```

ここでは、副プログラムADDの引数として値10を渡しています。この副プログラム内部で使われている変数Xは副プログラム内でのみ有効な局所変数として働き、副プログラムの外部で使われている変数Xとは別の変数として扱われます。
このため、100行で変数Xに代入された値10は120行まで継続され、120行で値10を表示します。

◆参照渡し

・副プログラムに変数を渡すと、渡された副プログラムはそれを参照として扱います。これを「参照によって渡す」といいます。
SUB行で宣言した副プログラムの引数への代入は、参照として渡された引数への代入となります。

例:

```
100 X=10
110 CALL ADD(X)
120 PRINT X
130 SUB ADD(A)
140  A=A+5
150 END SUB

＜実行結果＞
15
```

ここでは、副プログラムADDに変数Xを参照により渡しています。仮引数Aは変数Xの参照となり、140行の仮引数Aへの代入は変数Xへの代入となります。その結果として変数Xの値が15となります。

参照の利点

・参照を利用することで、参照している変数の内容を副プログラム内で変更することができます。

コラム3 副プログラムと参照

変数の値を渡す

・変数の値を引数に与えたい場合、引数に変数をそのまま書くと参照渡しとなってしまいます。変数を参照によって渡すのではなく、変数の中身の値を引数に与えたい場合、次の書式を用います。

――― 変数を()で囲う。

――― 変数に0を加算したり、1倍するなど引数の値に影響のない計算式を書く。

例:

```
100 X=10
110 CALL ADD((X))
120 PRINT X
130 SUB ADD(A)
140  A=A+5
150 END SUB

＜実行結果＞
10
```

ここでは、副プログラムADDの引数として変数Xの値を渡しています。このように変数をカッコ()で囲うことで、変数Xの値10を渡します。他の書き方として、CALL ADD(X+0)やCALL ADD(X*1)などがあげられます。いずれも引数を「式」にすることで、その結果である「値」を渡すという仕組みにしています。

参照の注意点

◆局所変数

・副プログラムで使っている仮引数は、局所変数として機能します。
局所変数は、その副プログラム内でだけ有効な変数で、副プログラムの終了とともに破棄されます。

・局所変数と同名の変数が、副プログラムの外部にある場合でも、副プログラム内では局所変数が使用されます。
そのため、同名の変数で使い道の違うふたつの変数が存在することになるので、混同しないよう気をつけてください。

・局所変数は、副プログラムの外部から使うことはできません。
もし局所変数と同名の変数を副プログラムの外部で使用した場合、それは新しい変数となります。

◆副プログラム内の変数

・副プログラム内で使用する、仮引数以外の変数は、広域変数となります。
広域変数は、その副プログラム内に限らず、外部からでも使用できる変数です。
副プログラム内部でのみ使用したい場合、副プログラムの外部で使用している変数と重複する変数名にならないよう気をつけてください。

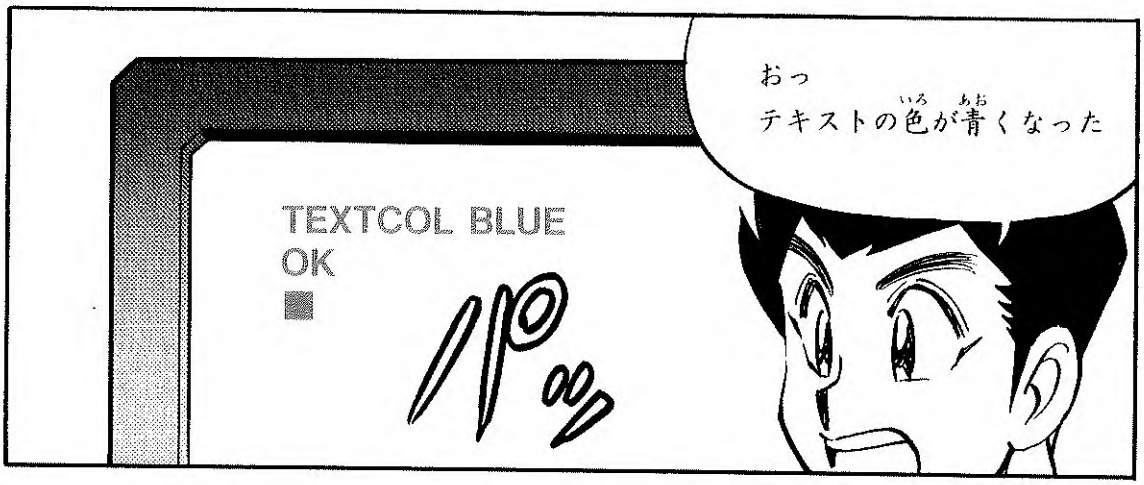
ゲーム作りへの挑戦

さて
オレが教えるのは
今日で最後だ
あれ？
もう終わりなの？

まあ基本的なことは
一通り教えたからな
最後はグラフィック関係を
教えて終わりにしよう

まずは簡単なところで
テキストの色を変えて
みよう
ありゃ…
またテキストか…
おっ！！ついに
グラフィックが
使えるんだな
やったー！

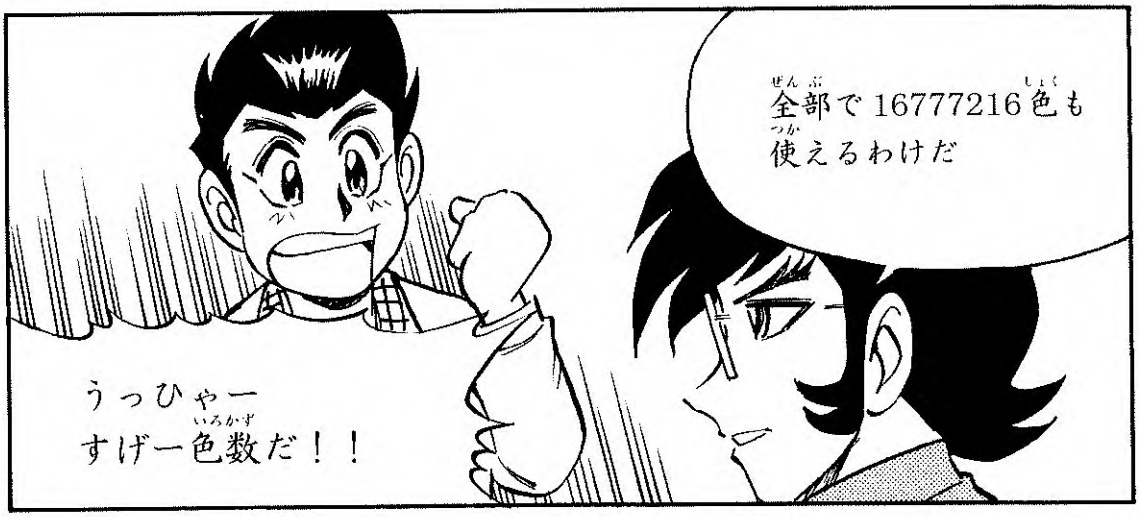
テキストの色を変えるには
TEXTCOL命令を使う
TEXTCOL BLUE■
こんな感じだな

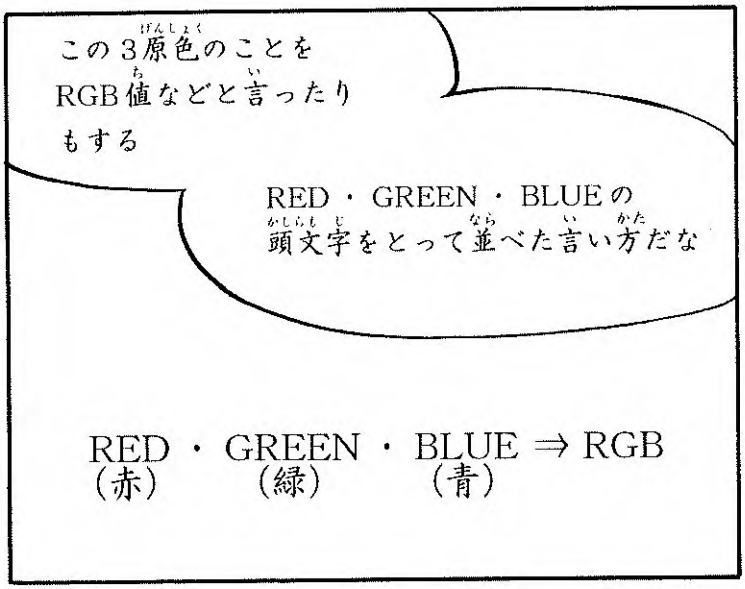

おっ
テキストの色が青くなった
TEXTCOL BLUE
OK
■
パッ

テキストには
これらの色が
使えるぞ
BLUE 青色
RED 赤色
GREEN 緑色
CYAN 水色
PURPLE 紫色
YELLOW 黄色
WHITE 白色
BLACK 黒色
なるほど
元に戻すときは
WHITEで白に
変えればいいんだ
そういうことだ
これは簡単だろ

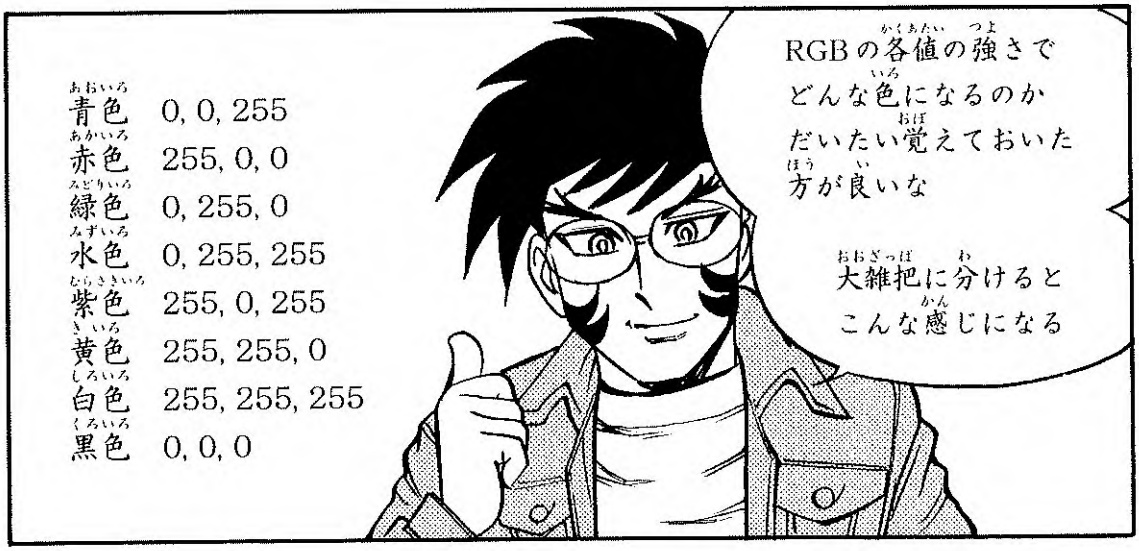
背景の色を変える
命令はBGCOLだ
TEXTCOL命令と
同じく8色で指定
するの？
次は文字の後ろ
背景の色を変える
命令だ
いや
この命令では光の3原色
赤・緑・青の色の強さを
0から255までの数値で
指定するんだ
こんな感じだ
BGCOL 255,0,0■
うおっ！
画面が真っ赤に
なったぜ
BGCOL 255,0,0
■

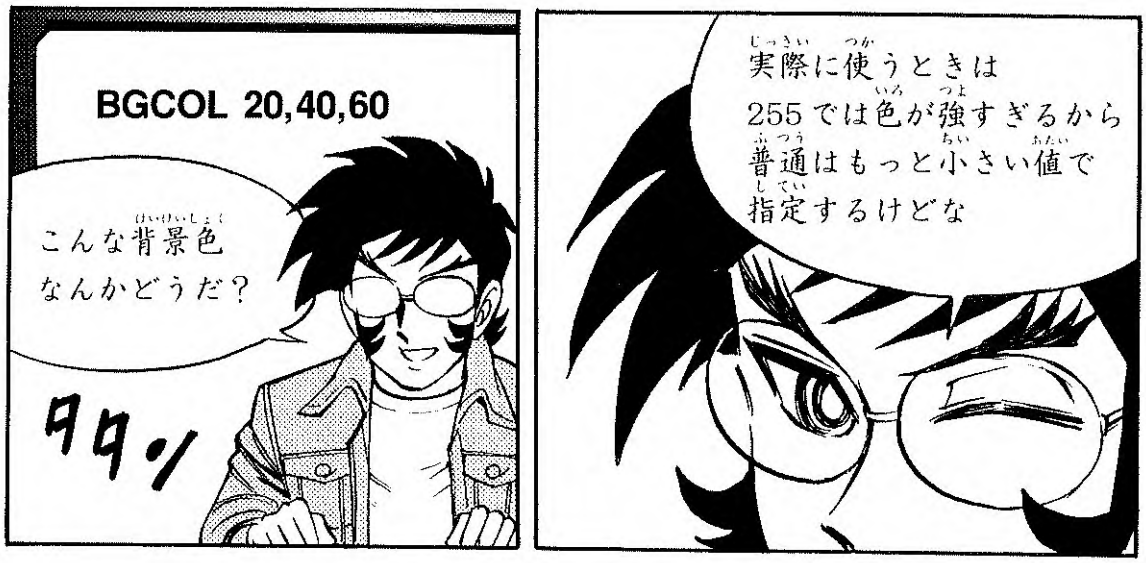
赤の強さを最大にして
BGCOL 255, 0, 0
緑と青の強さを
0にしたからな…
おっ
黒に戻ったぞ
ピー
タン
元の黒に戻すのは
こうだ
BGCOL 0,0,0
赤・緑・青の強さをそれぞれ
0から255の256段階で
指定できるってことは
合計で何色になると思う？
えーと
256段階が3つで…
256×256×256種類
かな…？
そうすると…


全部で16777216色も
使えるわけだ
うっひゃー
すげー色数だ！！


この3原色のことを
RGB値などと言ったり
もする
RED・GREEN・BLUEの
頭文字をとって並べた言い方だな
RED・GREEN・BLUE ⇒ RGB
(赤) (緑) (青)

グラフィック系の命令で
色の指定をするときは
主にこの3原色で行う

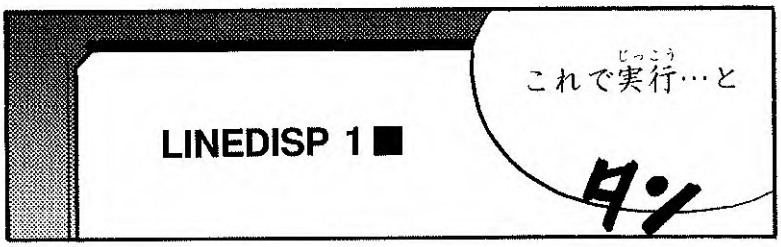

青色 0, 0, 255
赤色 255, 0, 0
緑色 0, 255, 0
水色 0, 255, 255
紫色 255, 0, 255
黄色 255, 255, 0
白色 255, 255, 255
黒色 0, 0, 0
RGBの各値の強さで
どんな色になるのか
だいたい覚えておいた
方が良いな
大雑把に分けると
こんな感じになる


BGCOL 20,40,60
こんな背景色
なんかどうだ？
実際に使うときは
255では色が強すぎるから
普通はもっと小さい値で
指定するけどな

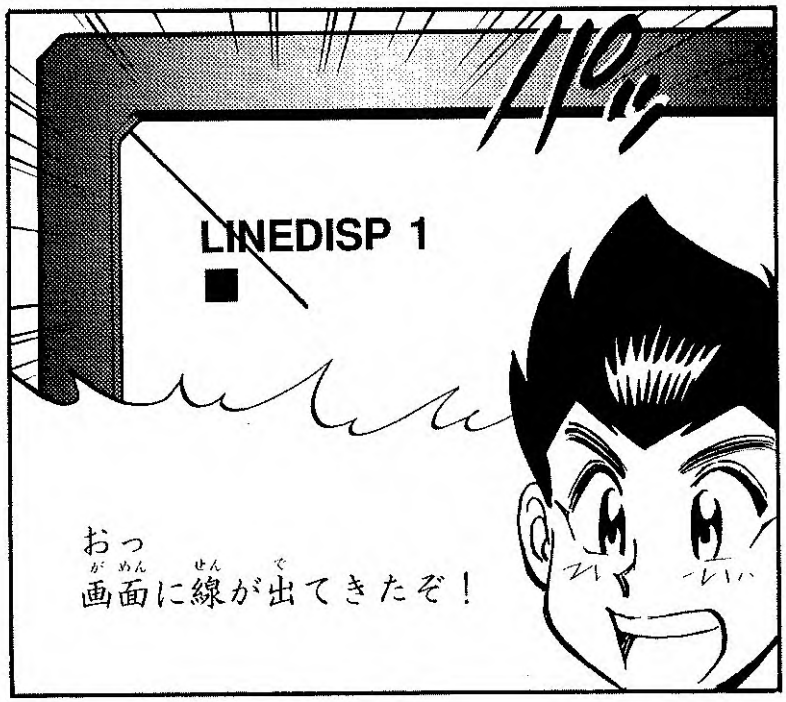
おっ 微妙に青くて
シブい背景色に
なったぜ
いろいろ試してみて
自分の好きな背景色で
作業をするのもいいだろう

線を表示してみよう
ライン…プリミティブ？
次はラインプリミティブを
使って線を表示してみよう

ラインプリミティブは
1本の直線を表示する
ための情報を持って
いるもので
グネグネと線を描いたときには
ラインプリミティブを幾つか
使って描いているんだ

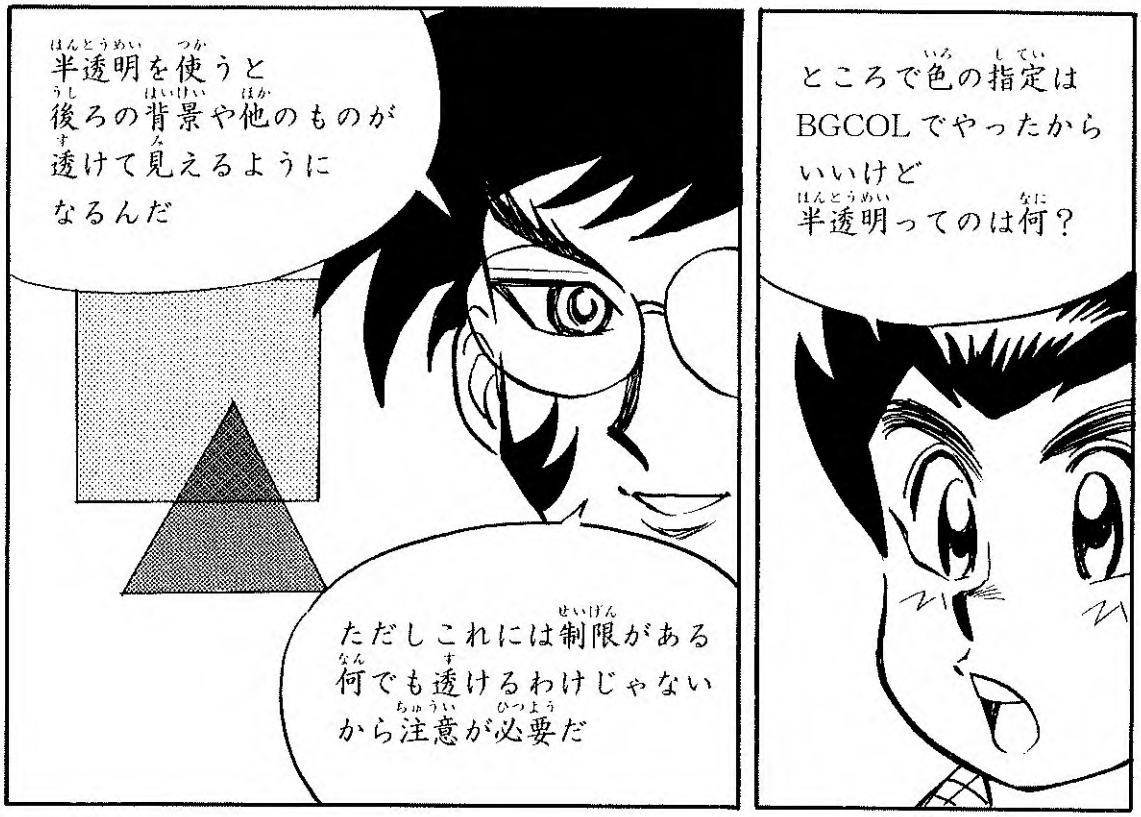

これで実行…と
LINEDISP 1■

ふーん
まあ
とりあえず
やってみよう


LINEDISP 1
■
おっ
画面に線が出てきたぞ！

LINEDISP 1
LINEDISP命令がラインプリミティブを表示させる命令だ
LINEDISP命令の後ろに付ける番号で何番目のラインプリミティブを表示するのか指定してやるんだ
つまりこの場合は1番目のラインプリミティブが表示されたわけだ
ふんふん…
それでラインプリミティブって何番まであるの？
1000番まで使えるぞつまり1000本の線が描けるってことだな
おおーなるほど

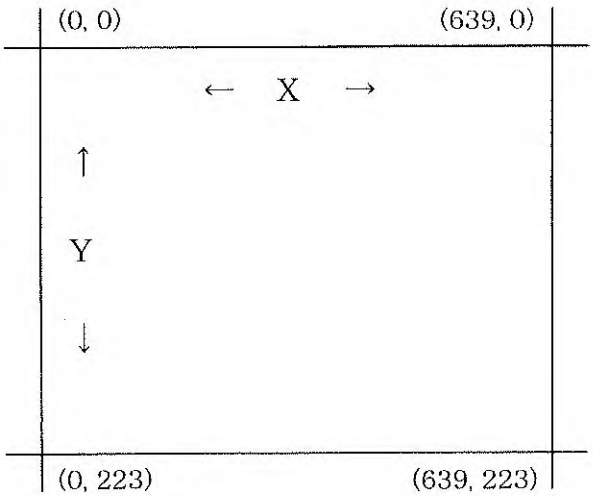
LINECOL命令で線の色を
変えることができる
ラインプリミティブの番号と
3原色で色を指定してやるんだ
次は表示した線の
色を変えてみよう
それと最後に半透明の指定を
0から128までの値でする
こんな感じだな
LINECOL 1, 255,0,0, 128■
おっ
線が赤くなった
半透明のことは
α(アルファ)値
ともいう
半透明の値は
0が完全に透明で
128が不透明だ


ところで色の指定は
BGCOLでやったから
いいけど
半透明ってのは何？
半透明を使うと
後ろの背景や他のものが
透けて見えるように
なるんだ
ただしこれには制限がある
何でも透けるわけじゃない
から注意が必要だ

ふーん
どんな制限なの？
まあ普通は不透明で
使うことが多いから
特に問題はないよ
うーん
気になるけど
了解だぜ
ちょっと複雑な
説明になるから
その話はパスだ

次はこの直線の
表示位置を
変えてみよう

ラインプリミティブの
表示位置を変更するのは
LINEPOS命令だ

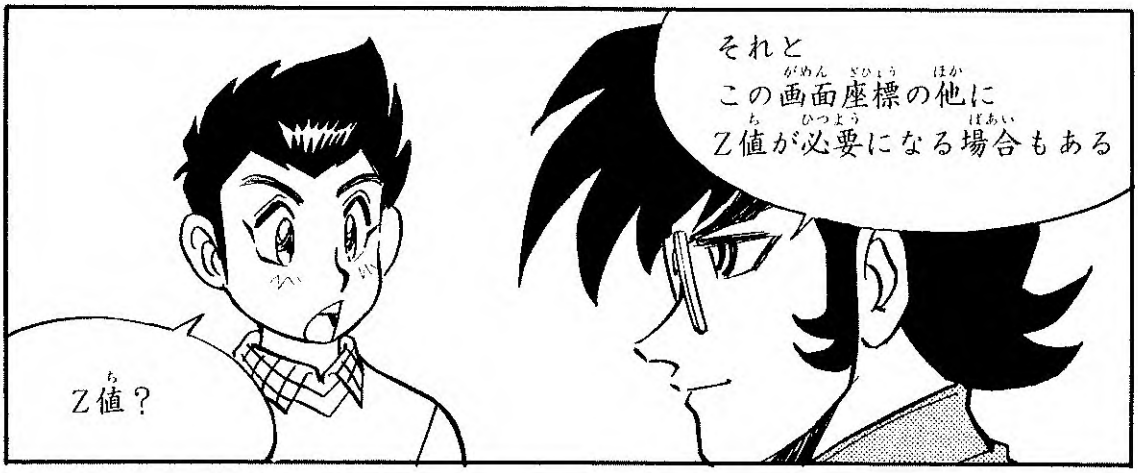


画面の座標は
XとYで指定する
左上が(0, 0)
右上が(639, 0)
左下が(0, 223)
右下が(639, 223)
になる

それと座標の値を見ると
横方向が縦方向より
細かく分かれている

座標1つ分の点を
ピクセルと呼ぶが
1つのピクセルの
形は縦長になる

ただしテレビに
よっては上下左右の
端の方が表示しきれ
ないものもあるから
注意だな

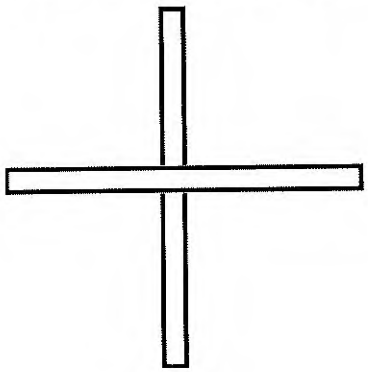

それと
この画面座標の他に
Z値が必要になる場合もある
Z値？

Z値とは奥行きを表すもので
Z値の大きいものほど
画面の手前に表示され
るんだ

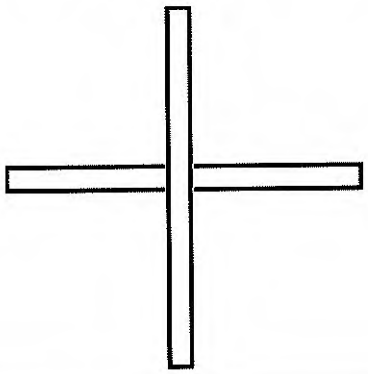
いやいや
拡大したり
縮小したりは
しないぞ
チチチッ

Z値を大きくすると
画面の手前に移動して
大きく見えるってこと？

横線のZ値が大きい場合



縦線のZ値が大きい場合



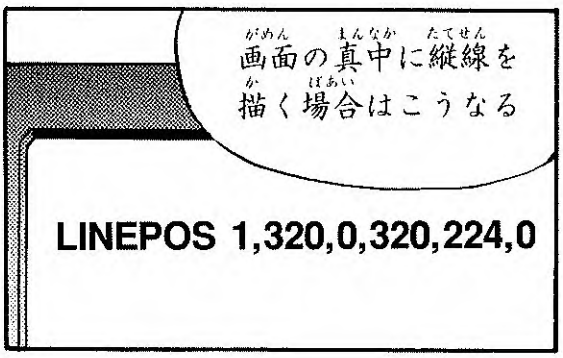
例えば
Z値の違う2本のラインが
交差していたとすると
Z値の大きい方が上に
表示されるってことだ

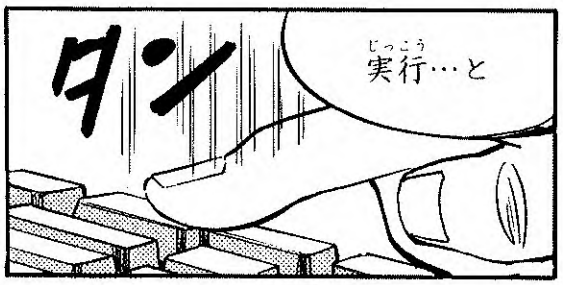
なるほどね
どっちを上に描くか
決めるための値なんだ

そういうことだ
まあ普通は0で
良いだろう

ふ～ん
了解

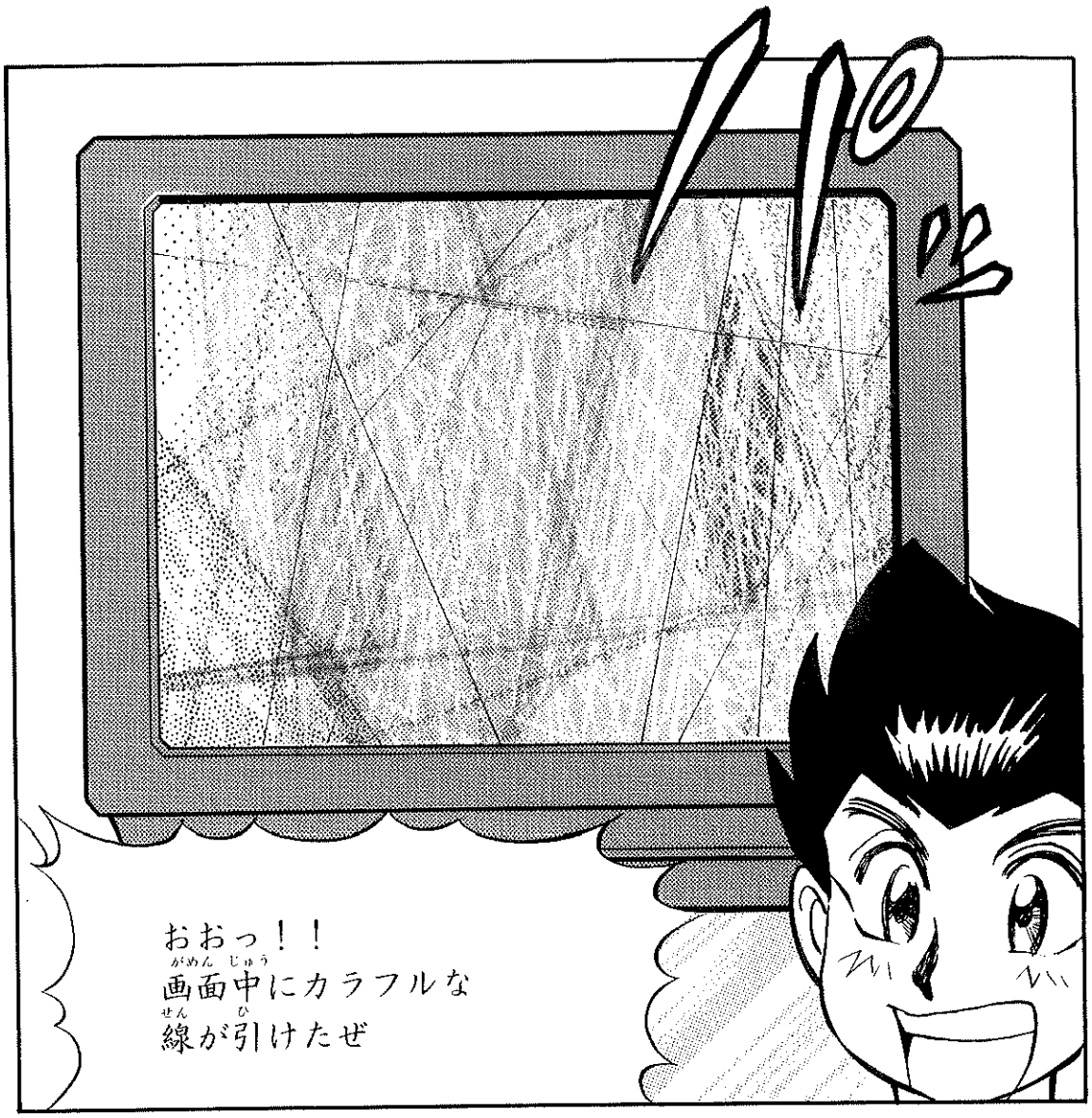
LINEPOS ライン番号，始点Ｘ，始点Ｙ，終点Ｘ，終点Ｙ，Ｚ値





※答えを見る前に挑戦してみよう

```
100 FOR I=1 TO 1000
110   LINECOL I, INT(RND * 256), INT(RND * 256),
INT(RND * 256), 128
120   LINEPOS I, INT(RND * 640), INT(RND * 224),
INT(RND * 640), INT(RND * 224), 0
130   LINEDISP I
140 NEXT I
■
```


ノイローゼ
おおっ！！
画面中にカラフルな
線が引けたぜ

でもこのままじゃ
画面が線だらけで
プログラムが読み難く
なったよ

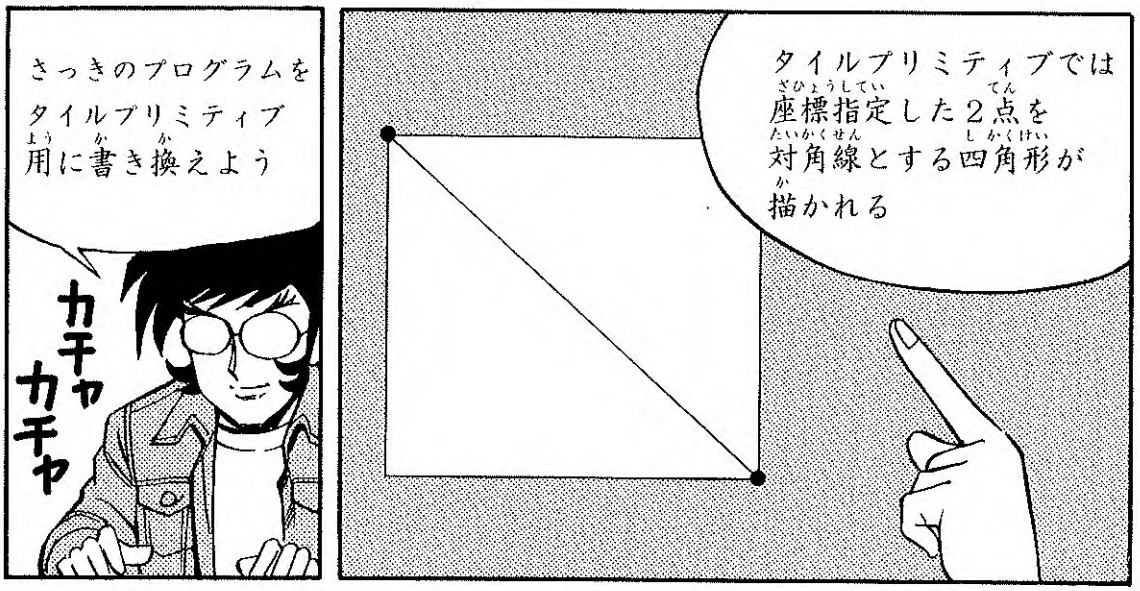
問題なく
できたようだな
へへーっ

おっ
元の画面に
もどった
そんなときは
INIT 命令だ
INIT ■
ターン
このINITは初期化用の命令で
色々な状態をすべて元に
もどしてくれるんだ
ふんふん
なるほど
ターン
ついでにもう1つ
教えておこう
TEXTMODE 2 ■
TEXTMODE 命令だ

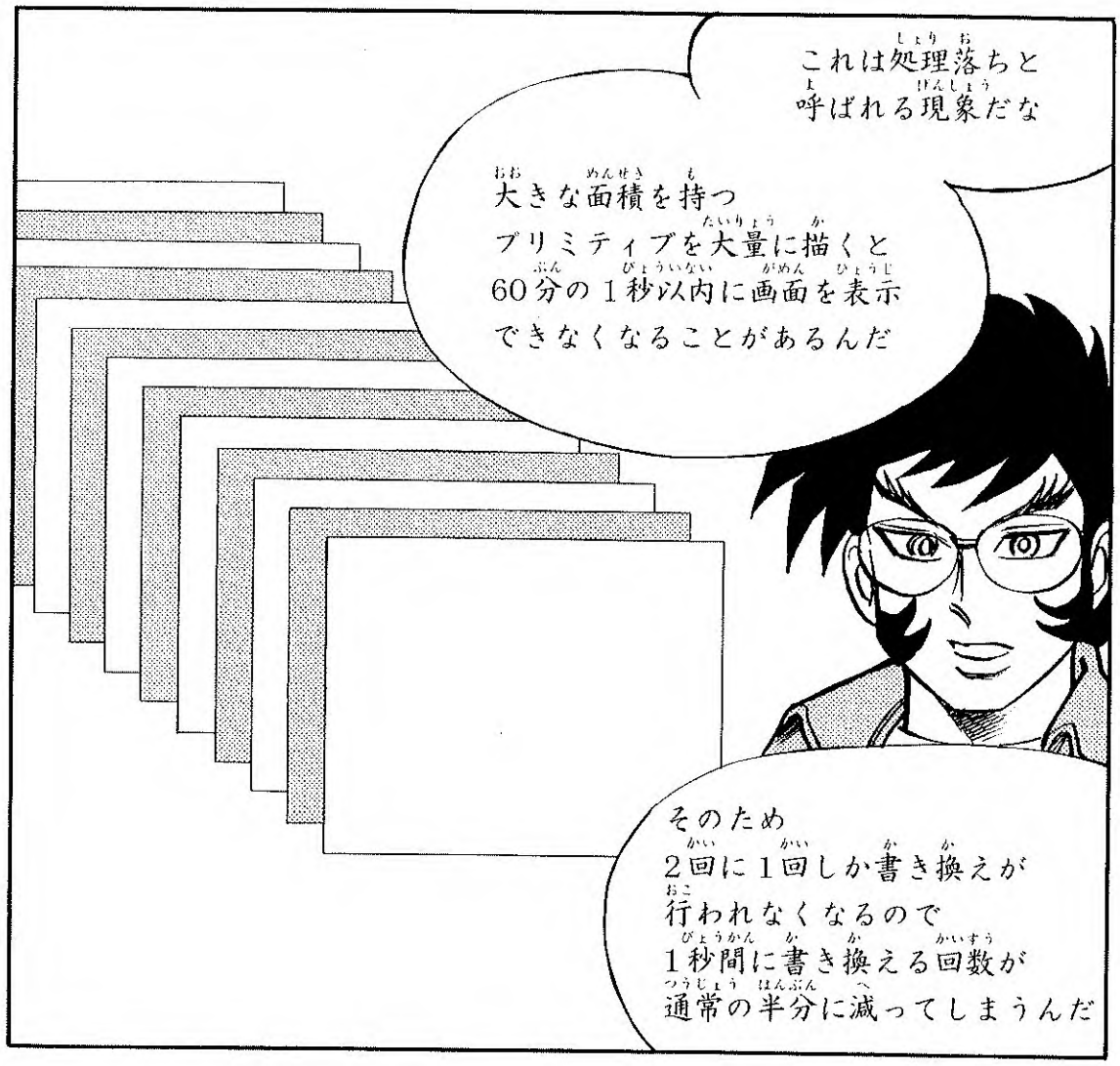
テキストには横38文字と
横50文字の2種類のモードがあるんだ
へーっ
OK
■
あっ！
文字が小さくなった
でも小さい文字はちょっと読みにくいな
読みにくかったら
元のテキストモードにもどせばいい
TEXTMODE 1■
TEXTMODE 1は横38文字
TEXTMODE 2は横50文字
で表示される
なるほど
了解
OK
■
おっ
元にもどった

LINEDISP ⇒ TILEDISP
LINEOFF ⇒ TILEOFF
LINECOL ⇒ TILECOL
LINEPOS ⇒ TILEPOS
線の次は四角形を描く命令を教えよう
四角形を描くのはタイルプリミティブだ
使い方はラインプリミティブと同じだな
命令のLINEがTILEに変わるだけだ

指定する項目もすべて同じだな
タイルプリミティブも1000番まで使えるぞ


さっきのプログラムをタイルプリミティブ用に書き換えよう
カキャ
カキャ
タイルプリミティブでは座標指定した2点を対角線とする四角形が描かれる

RUNで実行だ
ターン
100 FOR I=1 TO 1000
110 TILECOL I, INT(RND * 256), INT(RND * 256), INT(RND * 256), 128
120 TILEPOS I, INT(RND * 640), INT(RND * 224), INT(RND * 640), INT(RND * 224), 0
130 TILEDISP I
140 NEXT I
■
おっ
画面中にカラフルな
四角形が描けたぞ！
何か文字が荒くて
汚くなったぞ
それに何だかキー入力が
重くなった感じだ
ん…


これは処理落ちと
呼ばれる現象だな
大きな面積を持つ
プリミティブを大量に描くと
60分の1秒以内に画面を表示
できなくなることがあるんだ
そのため
2回に1回しか書き換えが
行われなくなるので
1秒間に書き換える回数が
通常の半分に減ってしまうんだ

それと日本のテレビは
NTSCという方式なので
走査線の偶数ラインと
奇数ラインを交互に表示
している

ふーん
なんとなく
分かったような…

NTSC？
走査線？
？
まあテレビについては
詳しく説明しないが
要は処理落ちすると画面が
汚く表示されるってことさ
処理落ちしたときは
片方のラインしか表示されなく
なるから画面が汚くなるんだ
INIT ■
カタ
カタ
ひとまず
INIT 命令で元にもどそう
プリミティブの面積が
小さければ問題はないから
大きな面積のプリミティブを
大量に描かないように
気を付けるしかない
処理落ちは
プログラムでは
どうにもならない
からな
ふ～ん
了解
おっ
元にもどった

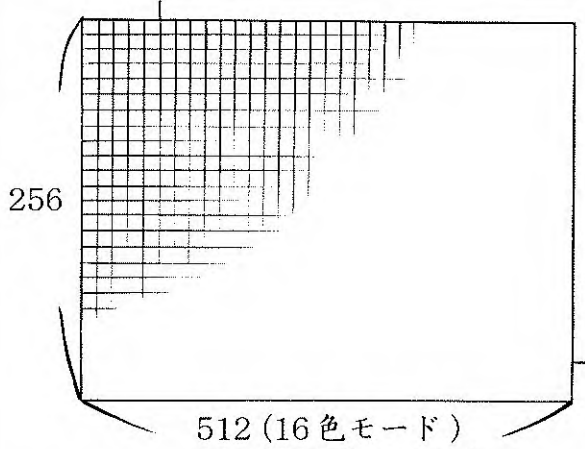
スプライトプリミティブ

次はスプライトプリミティブだ
これは今までのプリミティブ
よりちょっと複雑だぞ

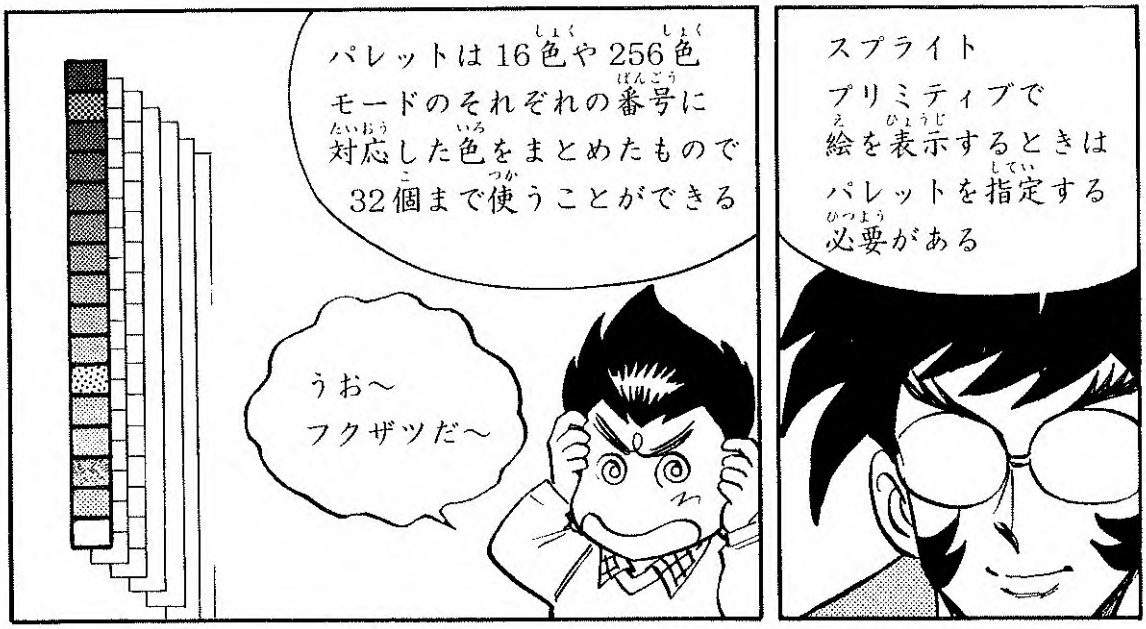
タイル
プリミティブは
このくらいに
しておこう

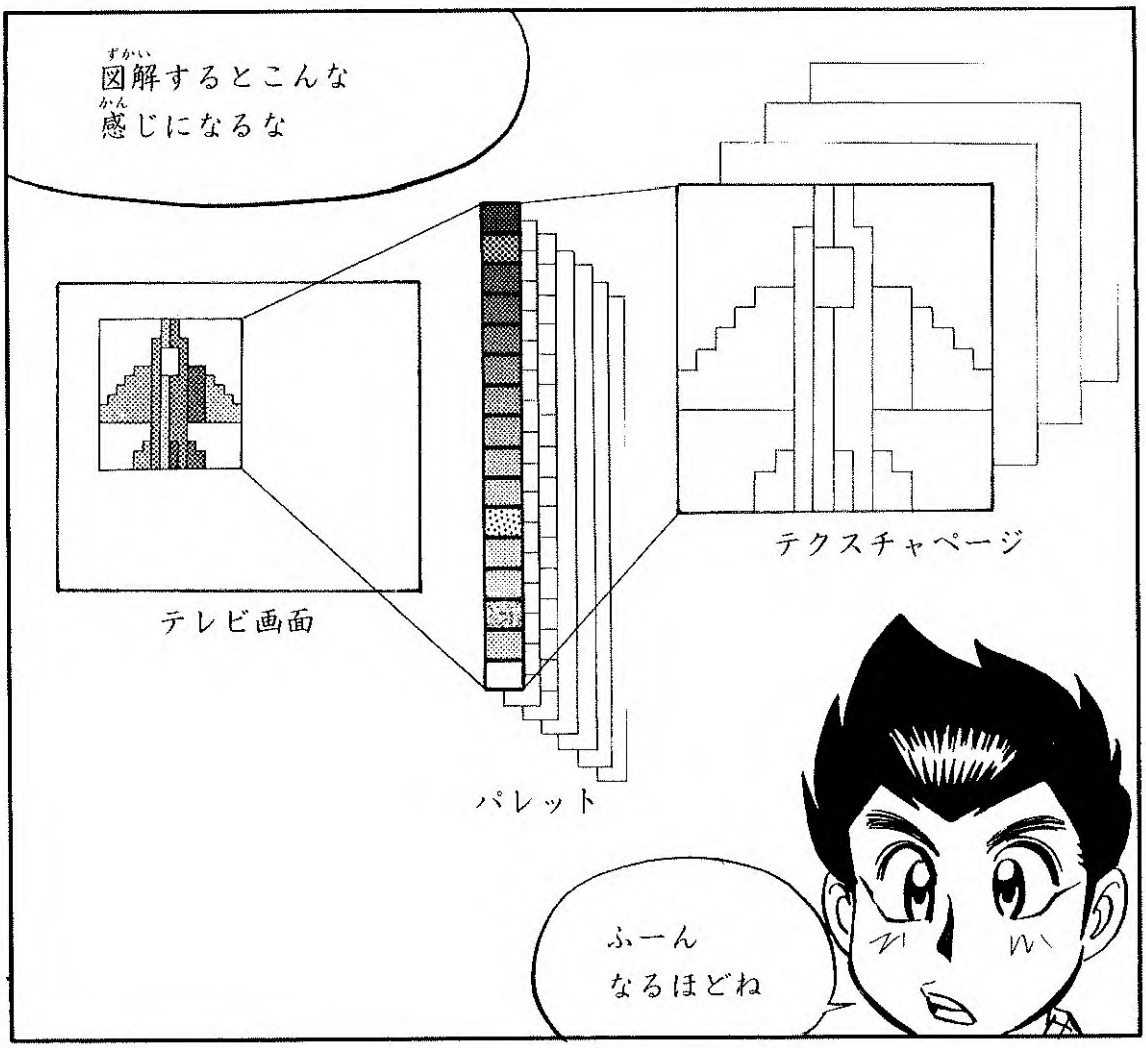
スプライトプリミティブは
テクスチャページの絵を
表示してくれる

テクスチャページってのは
絵のデータが保存されている
場所で16ページ分ある



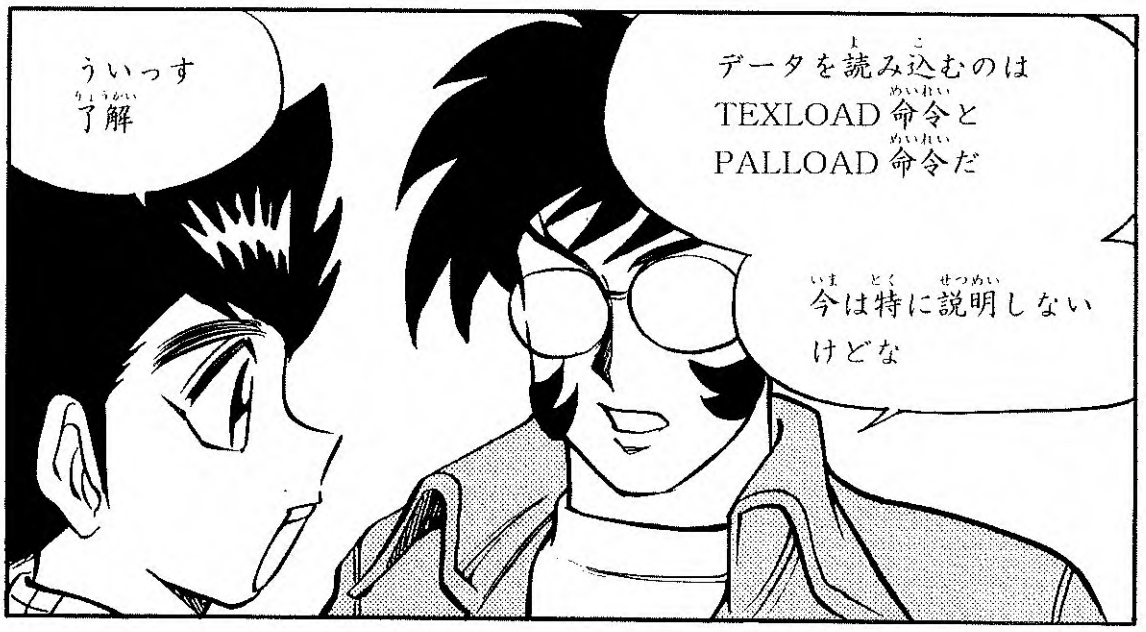
テクスチャページの絵は
16色か256色のどちらかで
1ページの大きさは
16色モードで512×256
256色モードで256×256
となる

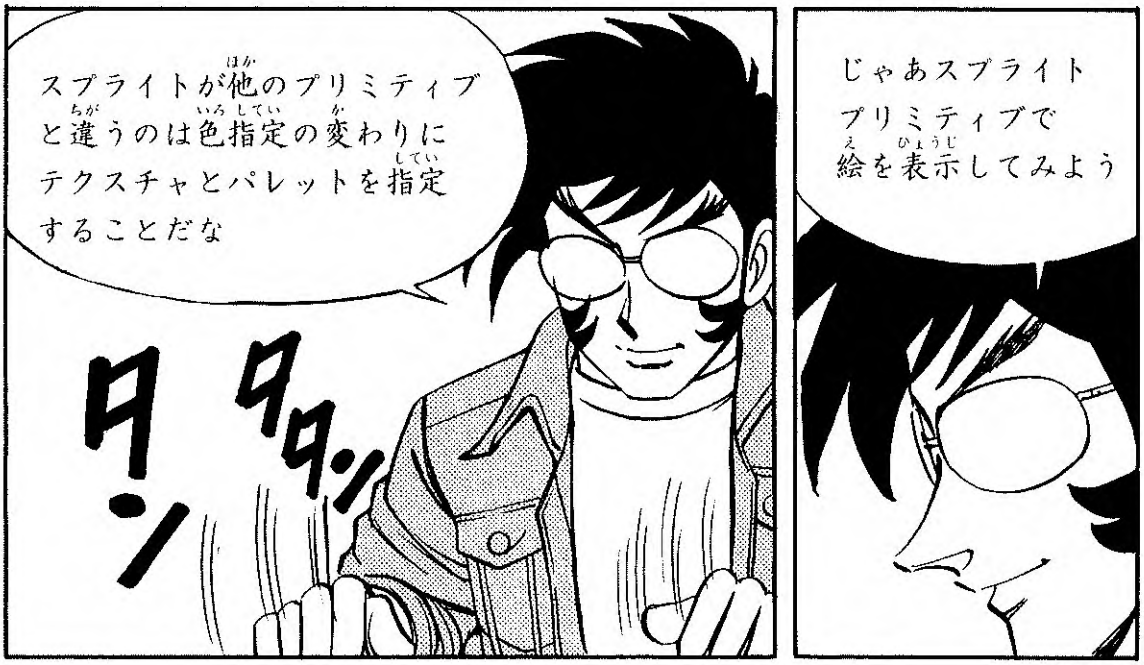

パレットは16色や256色
モードのそれぞれの番号に
対応した色をまとめたもので
32個まで使うことができる
うお〜
フクザツだ〜
スプライト
プリミティブで
絵を表示するときは
パレットを指定する
必要がある

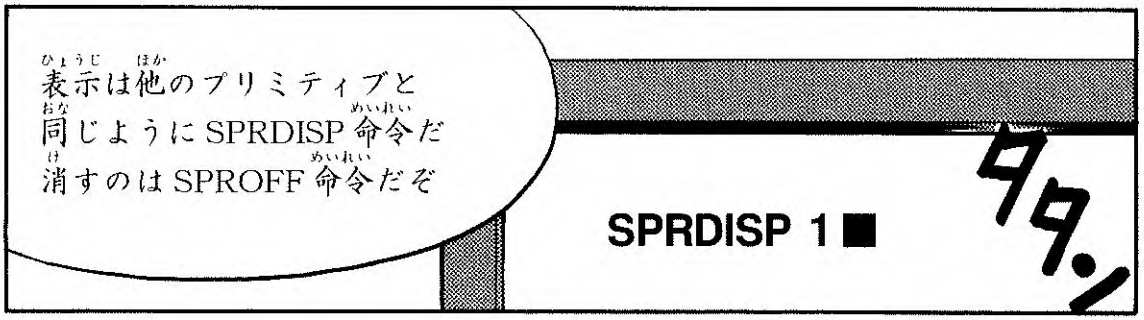

図解するとこんな
感じになるな
テレビ画面
パレット
テクスチャページ
ふーん
なるほどね

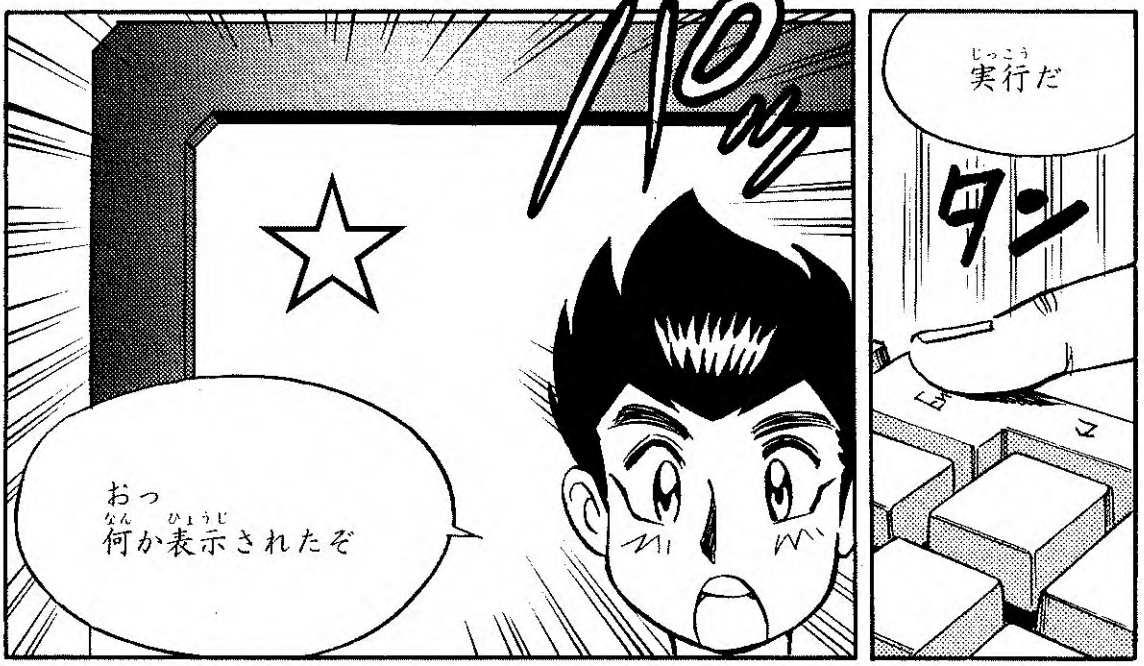
このテクスチャページの絵やパレットを作るにはIMAGE-SKETCHを使う
テクスチャページの絵やパレットはあらかじめ用意されているから使わなくても問題はないけどな
用意されているサンプル素材についてはマニュアルを見てくれ

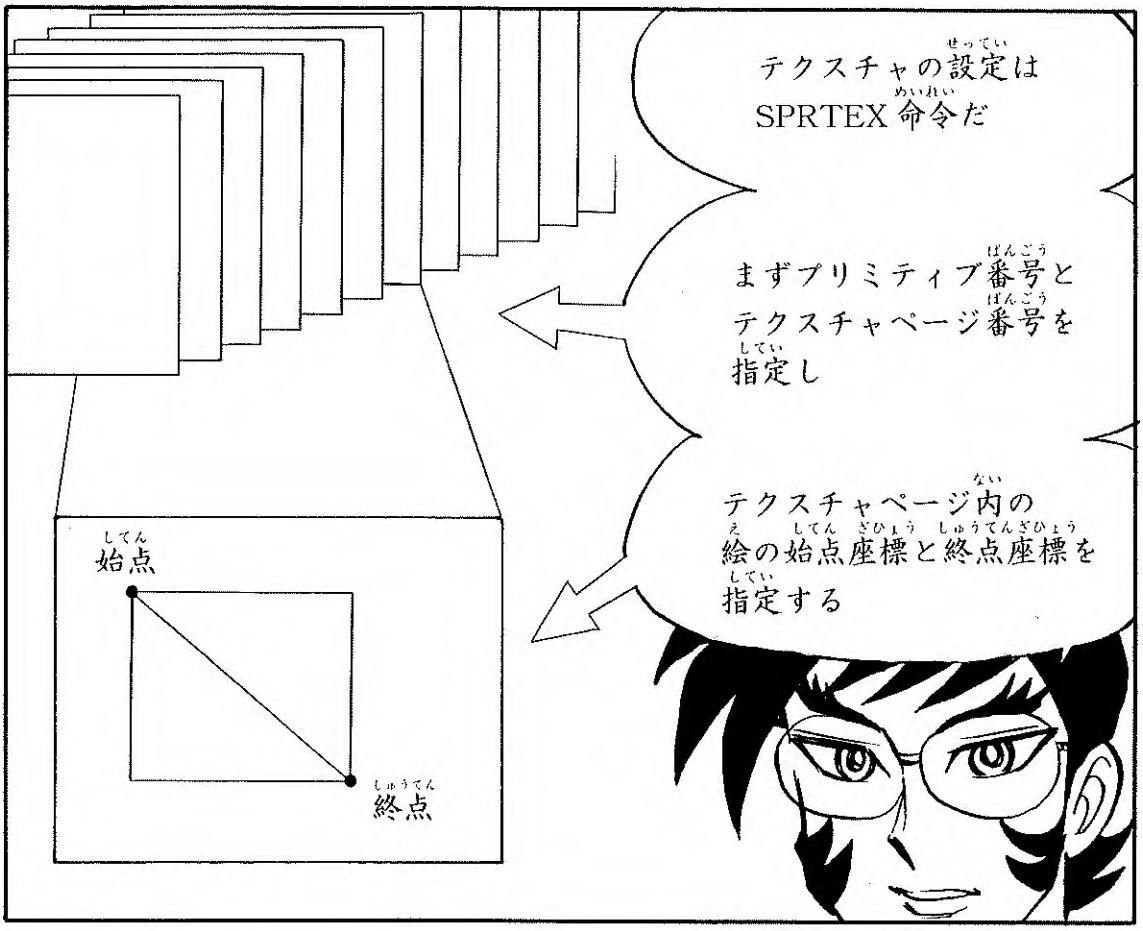
自分で作ったデータを読み込めばテクスチャページの絵やパレットを自由に変更できるぞ


データを読み込むのはTEXLOAD命令とPALLOAD命令だ
今は特に説明しないけどな
ういっす了解


スプライトが他のプリミティブと違うのは色指定の変わりにテクスチャとパレットを指定することだな
ウーン
じゃあスプライトプリミティブで絵を表示してみよう


表示は他のプリミティブと同じようにSPRDISP命令だ
消すのはSPROFF命令だぞ
SPRDISP 1■
タタン


ﾊﾟｰﾝ
おっ
何か表示されたぞ
実行だ
タン



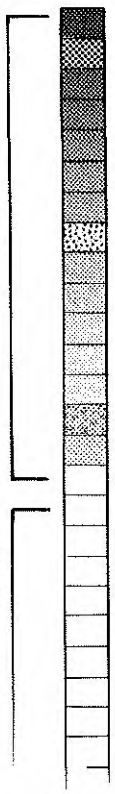
SPRTEX プリミティブ番号，テクスチャページ番号，
テクスチャ始点座標X，テクスチャ始点座標Y，
テクスチャ終点座標X，テクスチャ終点座標Y

続いてパレットの
指定はこうだ

SPRPAL プリミティブ番号，パレット番号，
オフセット値，アルファモード

パレット番号が
テクスチャページの絵と
合っていないと変な色の
スプライトが表示されるぞ

オフセット値が０の
ときは256色のうち
最初の16色が使われる



オフセットに１を足す
毎に16色分後ろに移動
した色が使われるんだ

1つのパレットは
256色分あるから
16色のテクスチャを
使用しているときは
オフセット値を指定
するんだ

オフセット値？

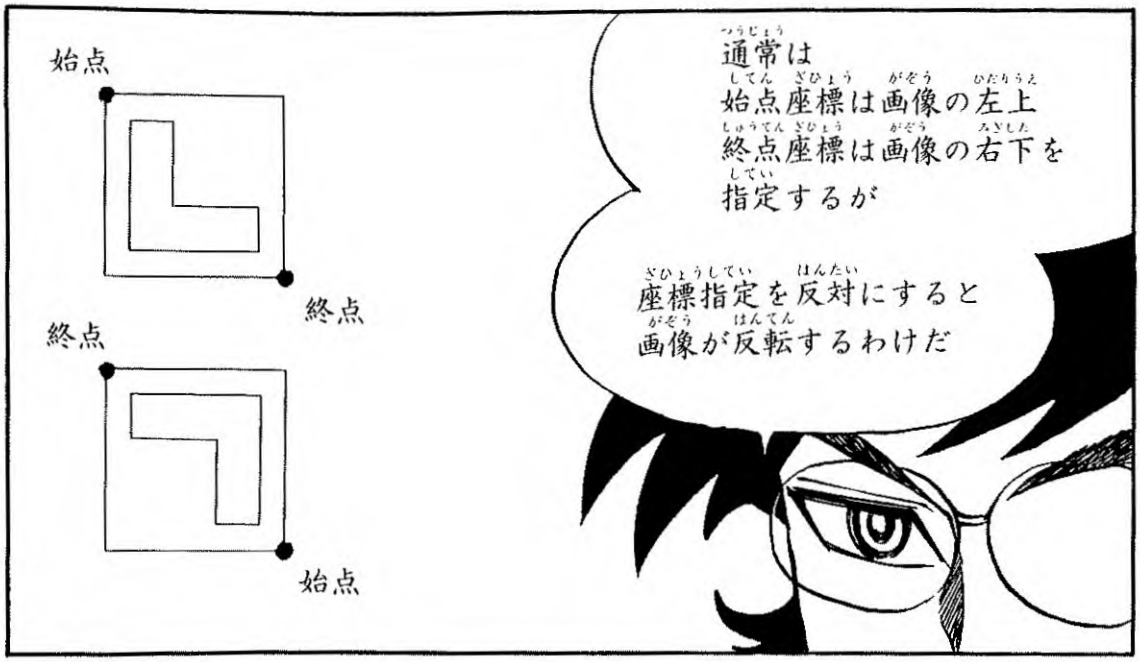
半透明を使用したときのモードだ
アルファモードは通常1で使用する
なるほど…でアルファモードってのは？
2を指定するとRGB値が加算されて表示される
0を指定したときは逆にRGB値が減算されるんだ
SPRPAL □,□,□,2 ⇒ 加算
SPRPAL □,□,□,0 ⇒ 減算
ああなるほど
要するに透ける色が明るくなったり暗くなったりするってことだ
RGB値？加算…減算？

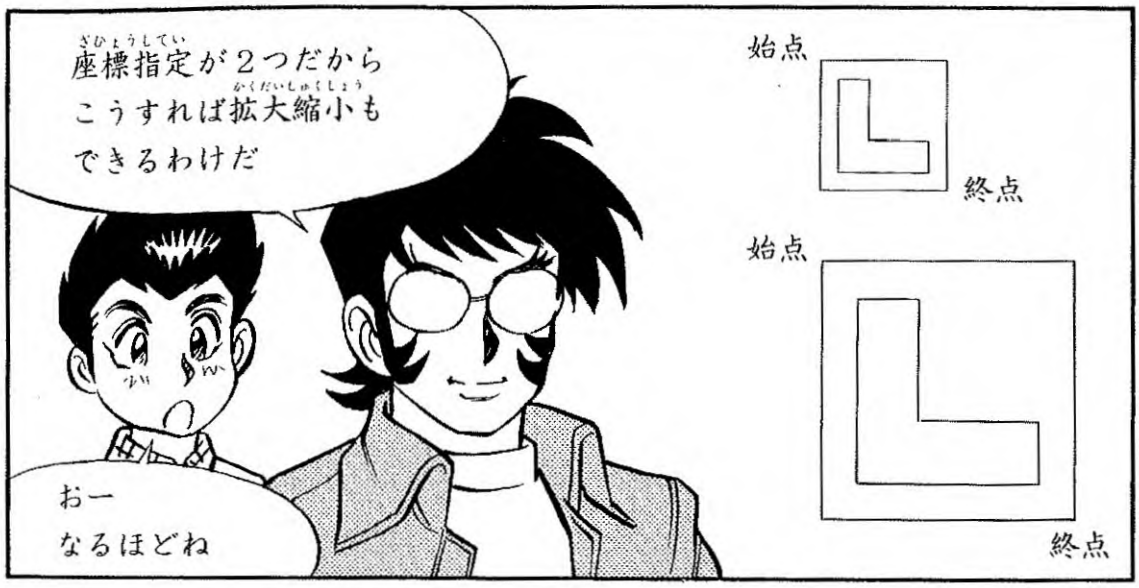
あとは画面での表示位置を
SPRPOS命令で指定する

SPRPOS プリミティブ番号，画面座標X，画面座標Y，Z値

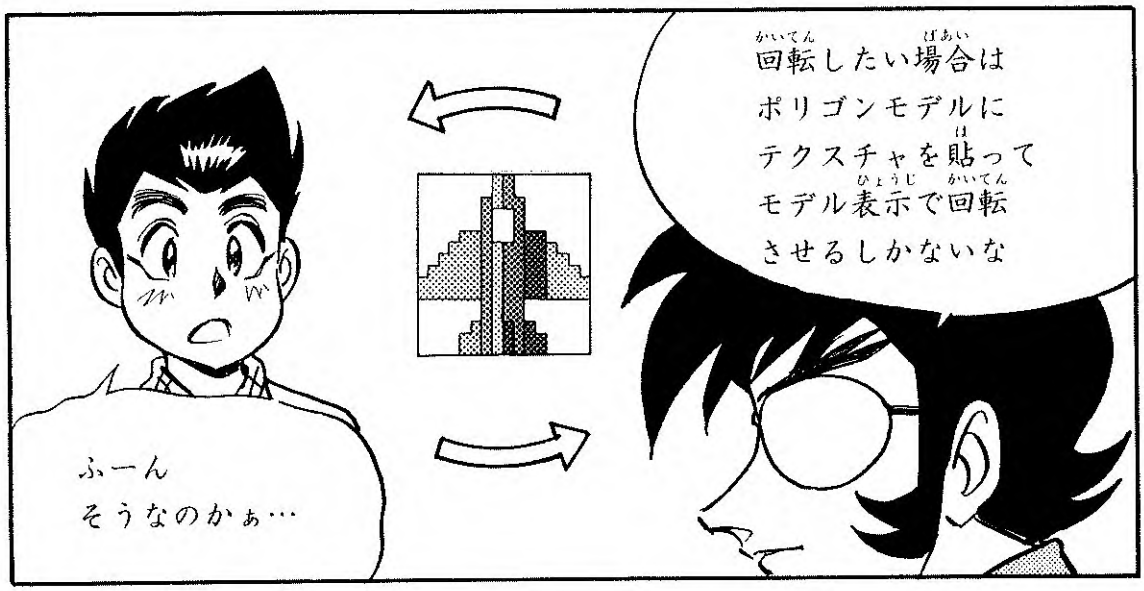
拡大縮小や反転などを行う
場合はSPRPOS2命令を使う
この命令では画面座標の
指定は2つになる

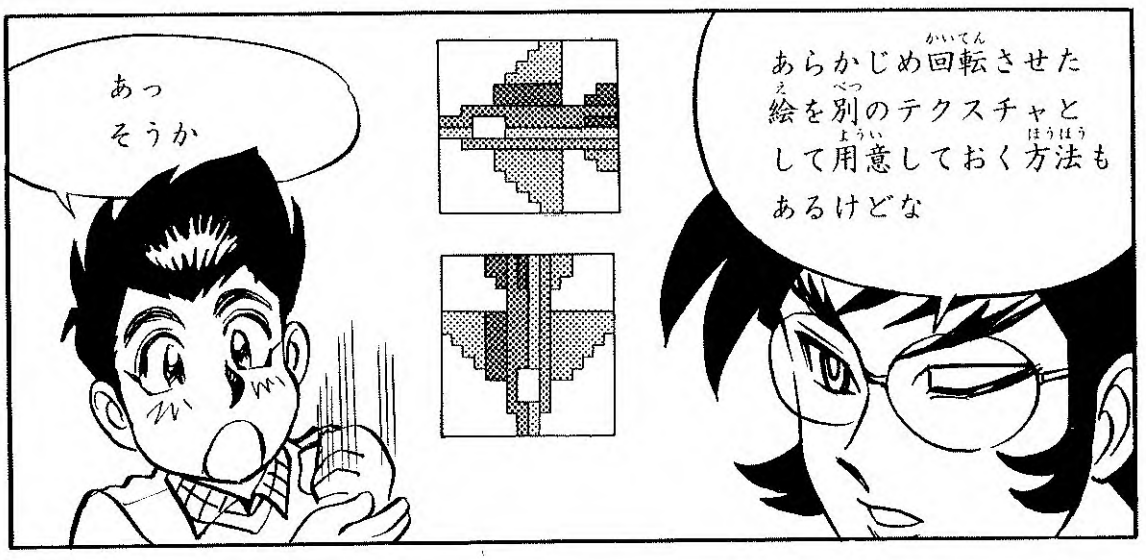
SPRPOS2 プリミティブ番号，始点座標X，始点座標Y，
終点座標X，終点座標Y，Z値

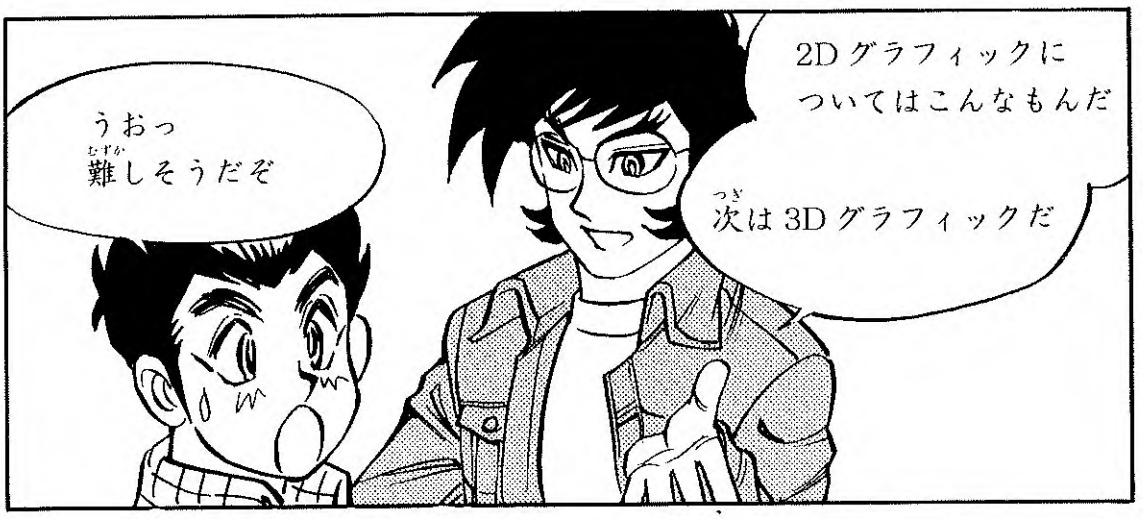

始点
終点
終点
始点
通常は
始点座標は画像の左上
終点座標は画像の右下を
指定するが
座標指定を反対にすると
画像が反転するわけだ

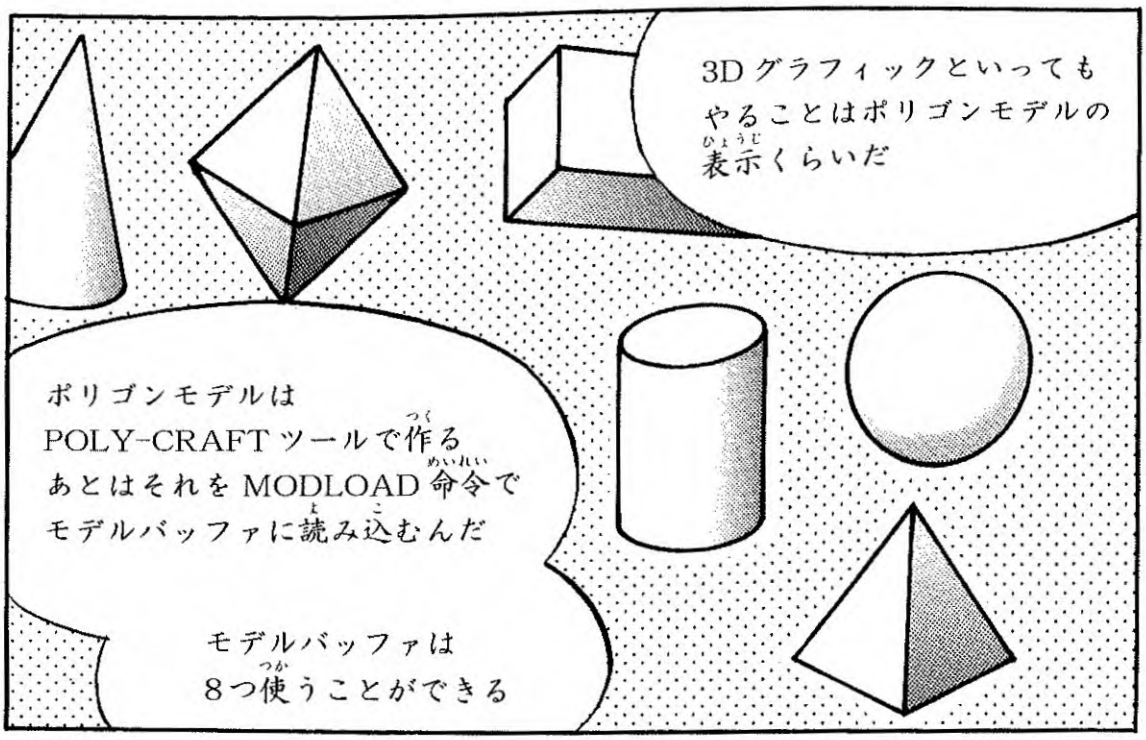

座標指定が2つだから
こうすれば拡大縮小も
できるわけだ
始点
終点
始点
終点
おー
なるほどね

回転は
どうやるの？
スプライト
プリミティブで
回転はできない

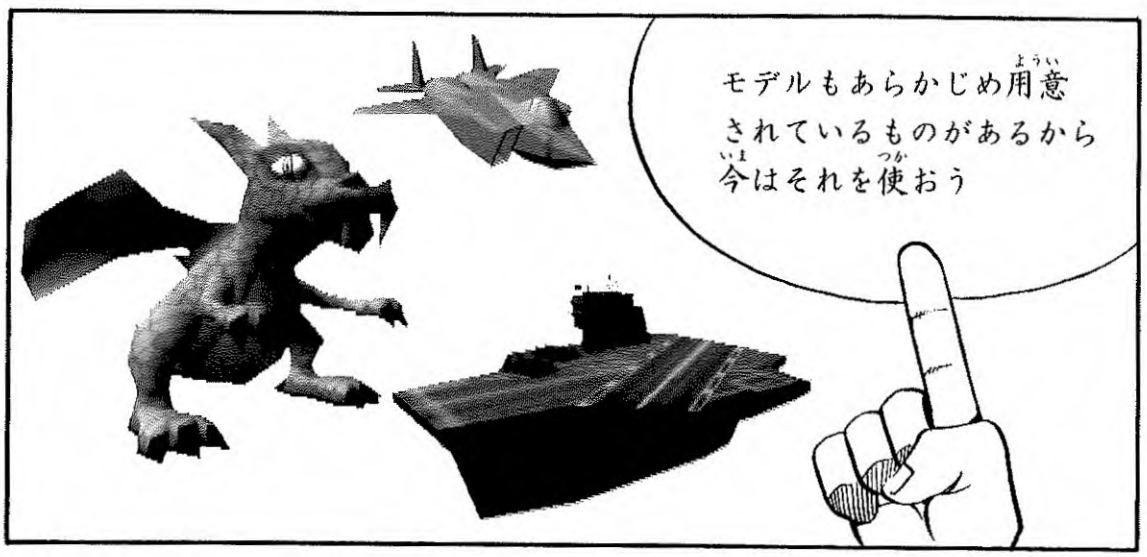

回転したい場合は
ポリゴンモデルに
テクスチャを貼って
モデル表示で回転
させるしかないな
ふーん
そうなのかぁ…

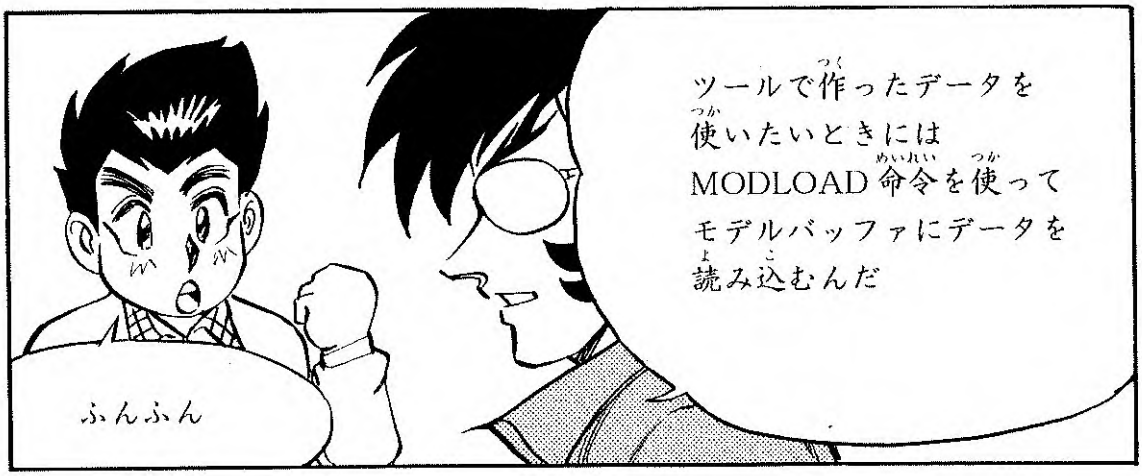

あらかじめ回転させた
絵を別のテクスチャと
して用意しておく方法も
あるけどな
あっ
そうか

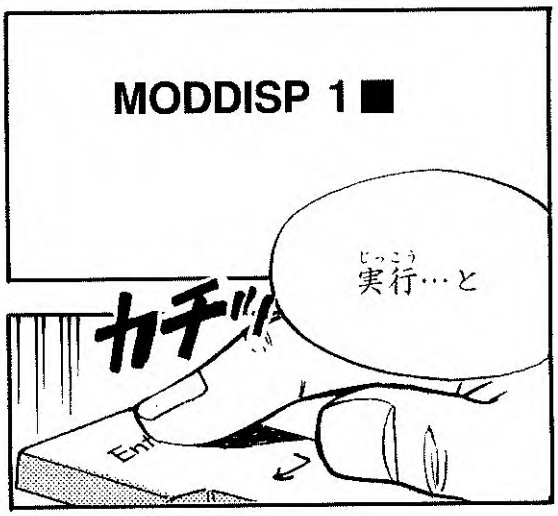

2Dグラフィックに
ついてはこんなもんだ
次は3Dグラフィックだ
うおっ
難しそうだぞ


3Dグラフィックといっても
やることはポリゴンモデルの
表示くらいだ
ポリゴンモデルは
POLY-CRAFTツールで作る
あとはそれをMODLOAD命令で
モデルバッファに読み込むんだ
モデルバッファは
8つ使うことができる

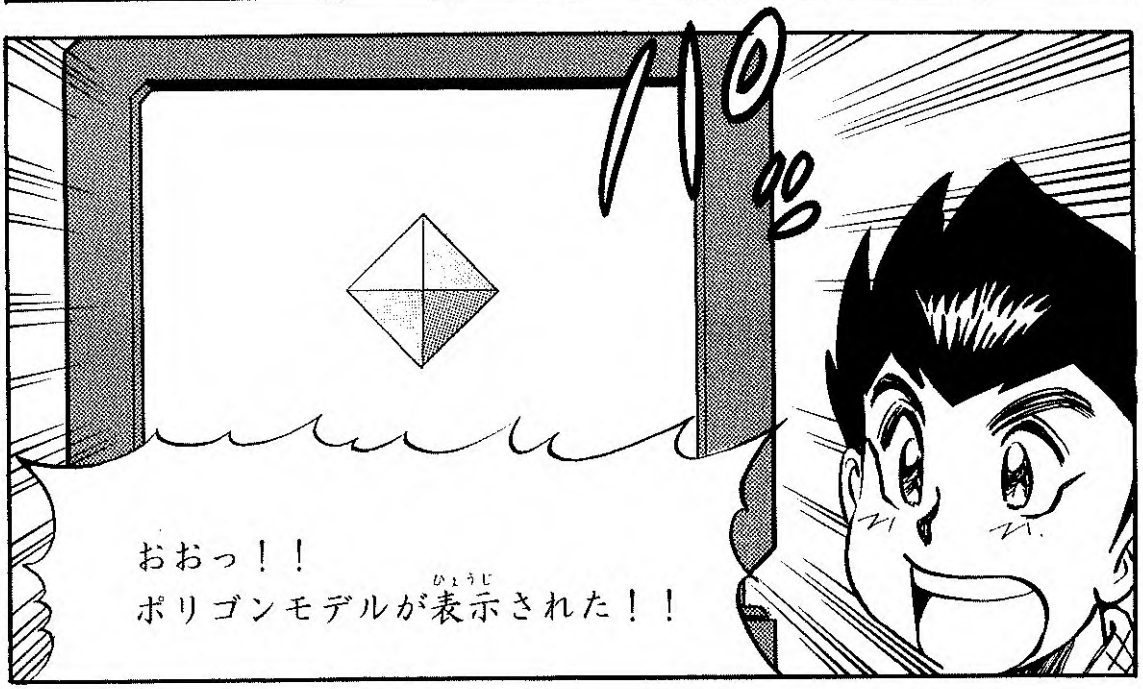
モデルにテクスチャを使って
いる場合には
テクスチャページの絵や
パレットも読み込んでおく
必要がある

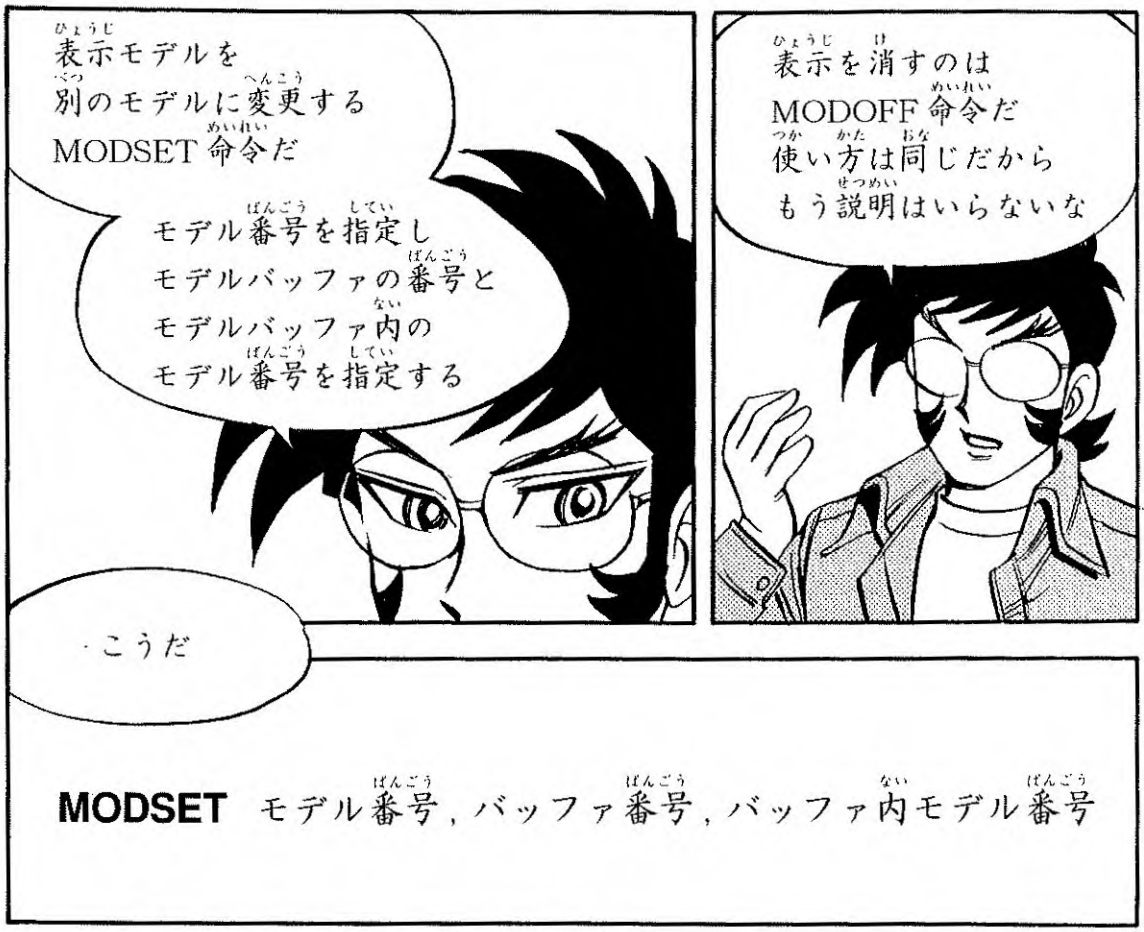

モデルもあらかじめ用意
されているものがあるから
今はそれを使おう

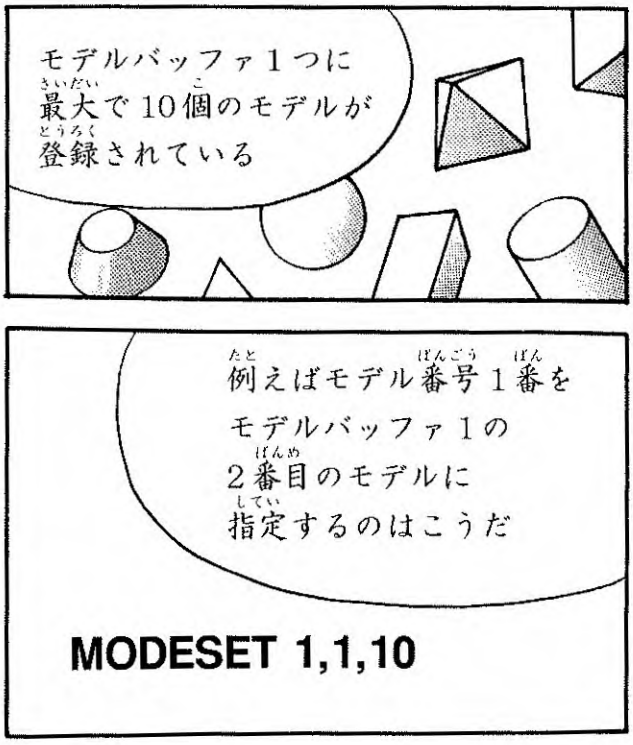

ツールで作ったデータを
使いたいときには
MODLOAD命令を使って
モデルバッファにデータを
読み込むんだ
ふんふん


MODDISP 1■
実行…と
カチッ
Enter

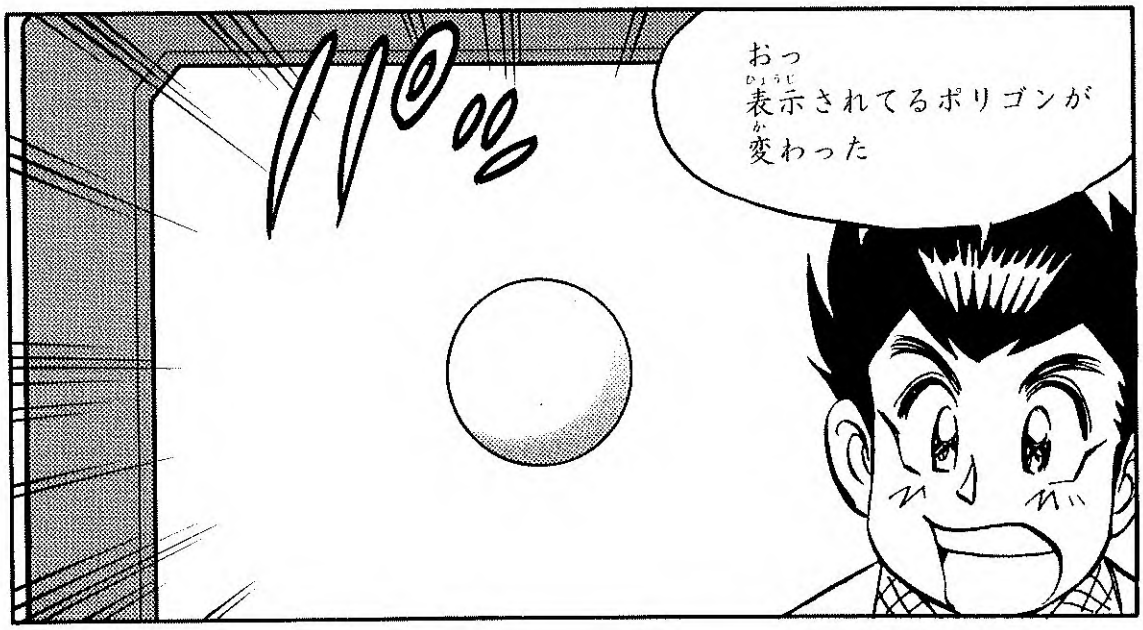
モデルの表示は
MODDISP命令だ
プリミティブと同じように
1000番まで使えるぞ

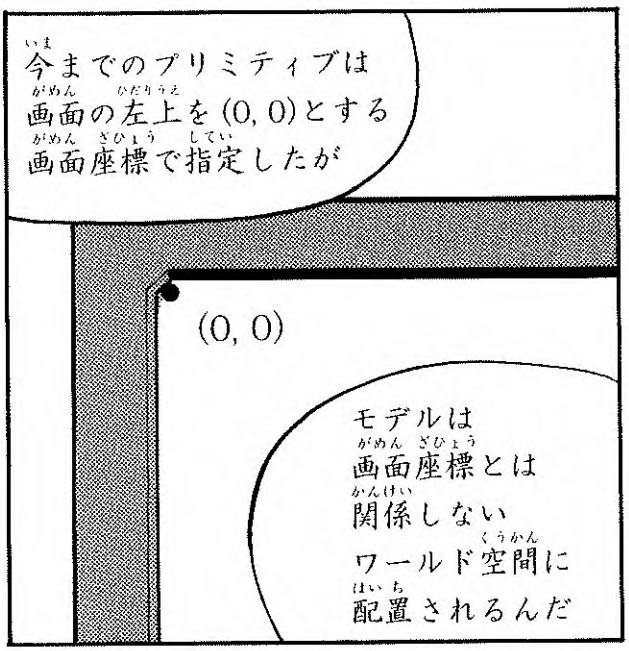

おおっ！！
ポリゴンモデルが表示された！！


表示モデルを
別のモデルに変更する
MODSET命令だ
モデル番号を指定し
モデルバッファの番号と
モデルバッファ内の
モデル番号を指定する
こうだ
表示を消すのは
MODOFF命令だ
使い方は同じだから
もう説明はいらないな
MODSET モデル番号，バッファ番号，バッファ内モデル番号


モデルバッファ１つに
最大で10個のモデルが
登録されている
例えばモデル番号１番を
モデルバッファ１の
２番目のモデルに
指定するのはこうだ
MODESET 1,1,10

スプライトの表示では
テクスチャページの絵を
表示したわけだが
モデル表示は
モデルバッファ内の
モデルを表示するわけだ
ふむ
ふむ

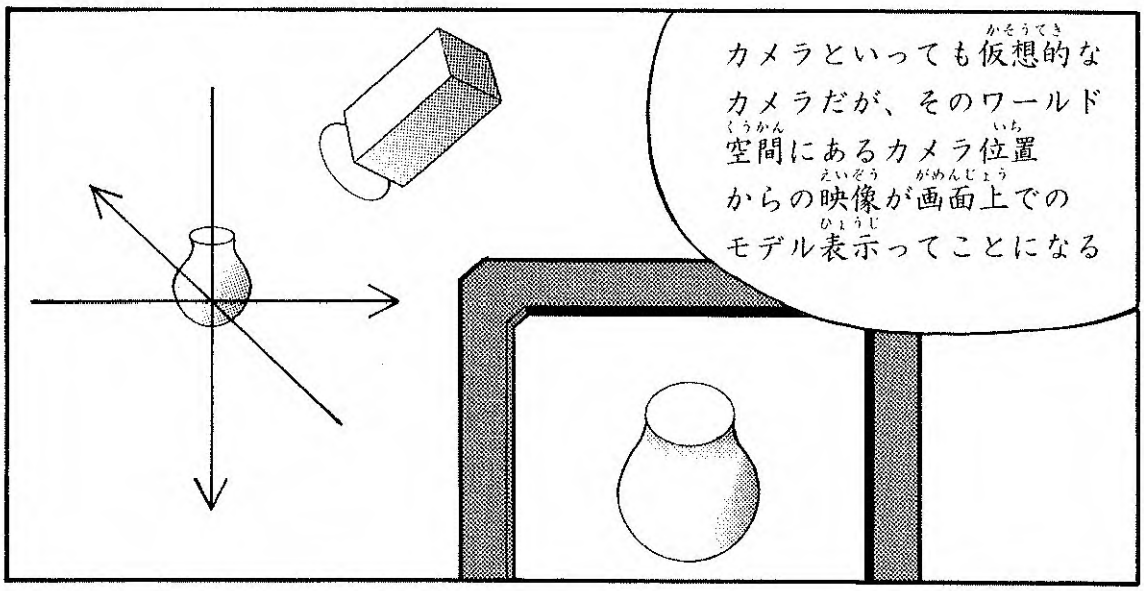

おっ
表示されてるポリゴンが
変わった

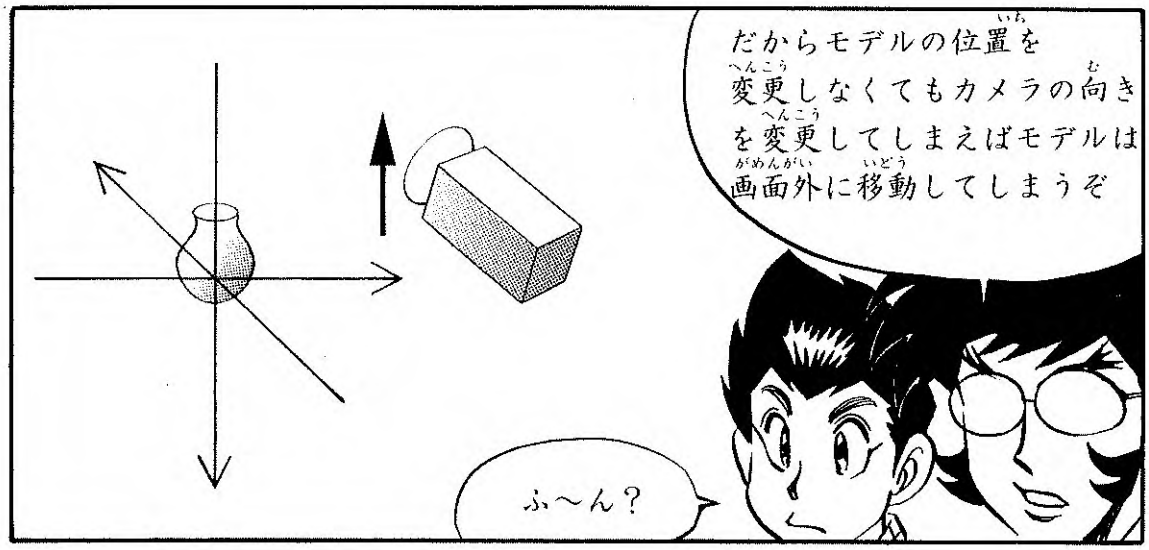

今までのプリミティブは
画面の左上を(0, 0)とする
画面座標で指定したが
(0, 0)
モデルは
画面座標とは
関係しない
ワールド空間に
配置されるんだ

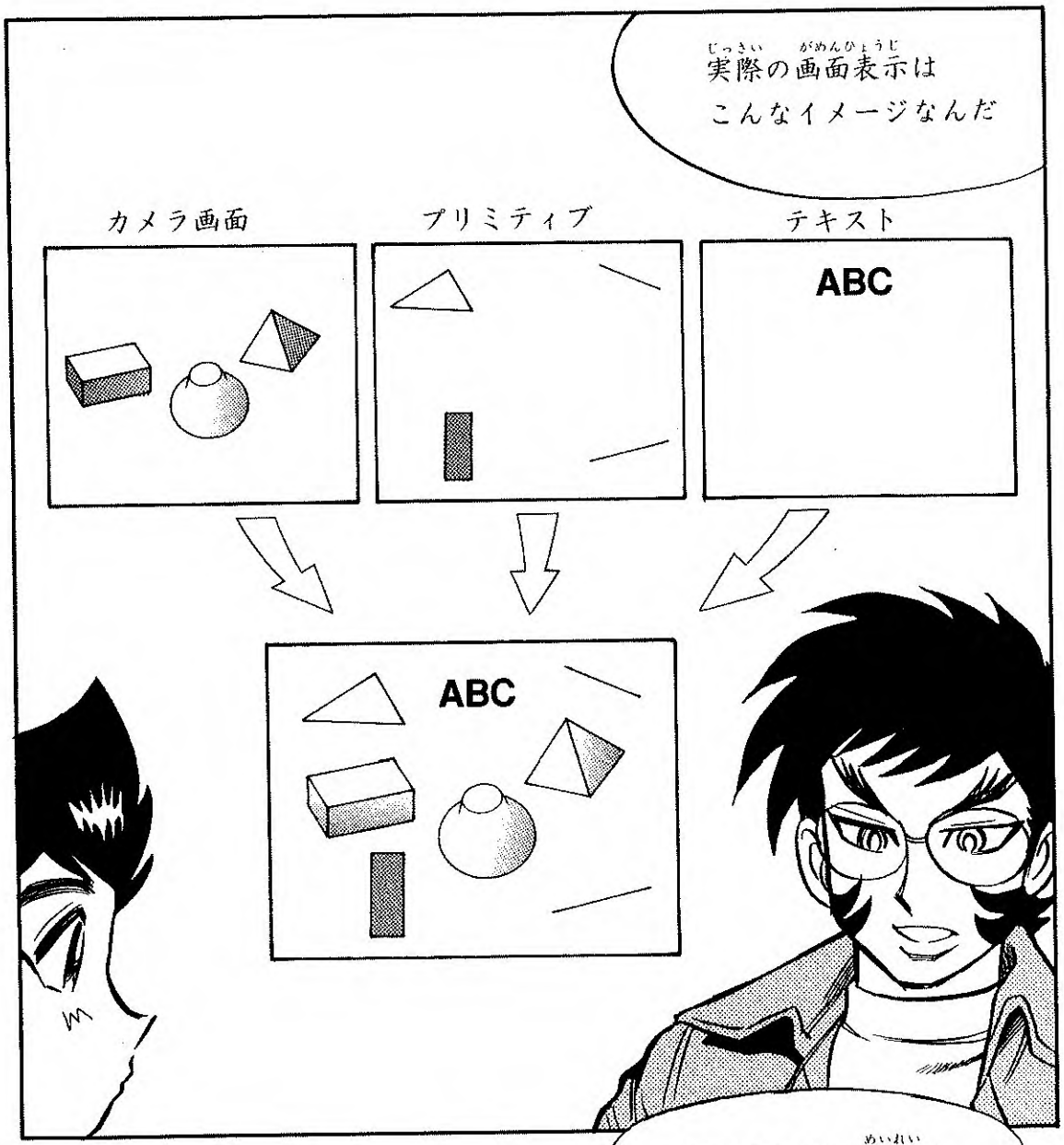
モデルの位置を変更する
場合はMODPOS命令だ
プリミティブと違って
ワールド空間座標で
指定する
ワールド空間座標？

ワールド空間ってのは
何なんだ？

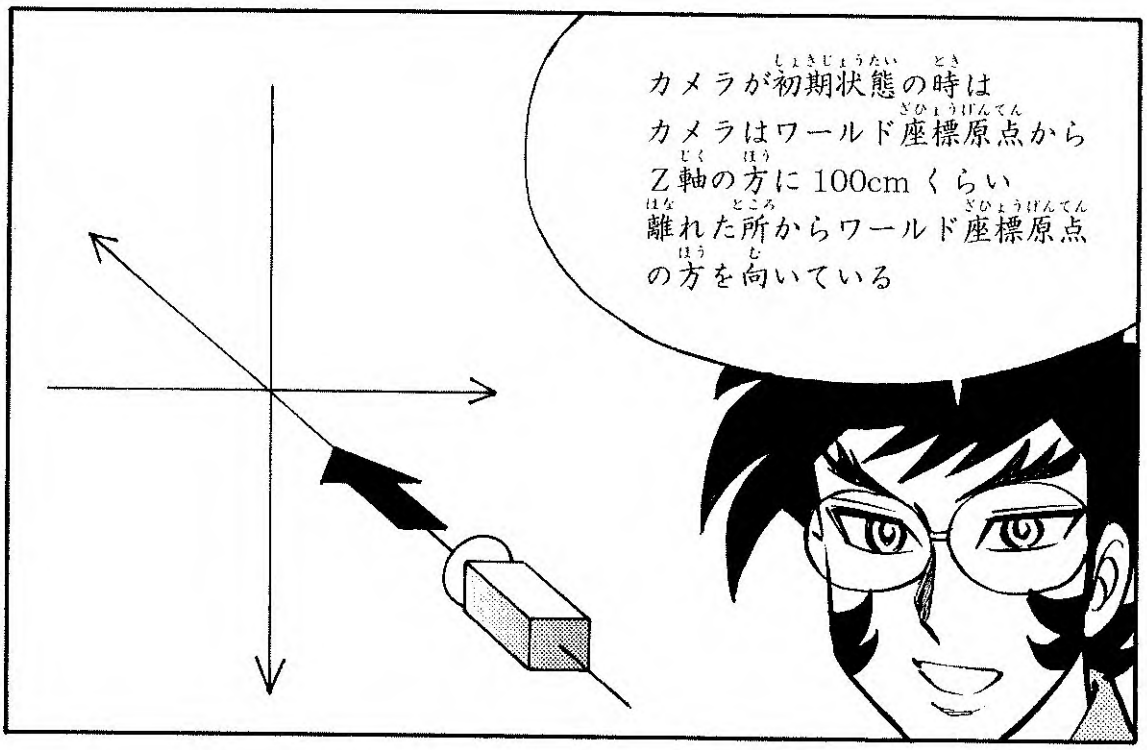
モデルを配置するための空間さ
画面に表示されているモデルは
ワールド空間に配置された
カメラからの映像なんだ
カメラ？

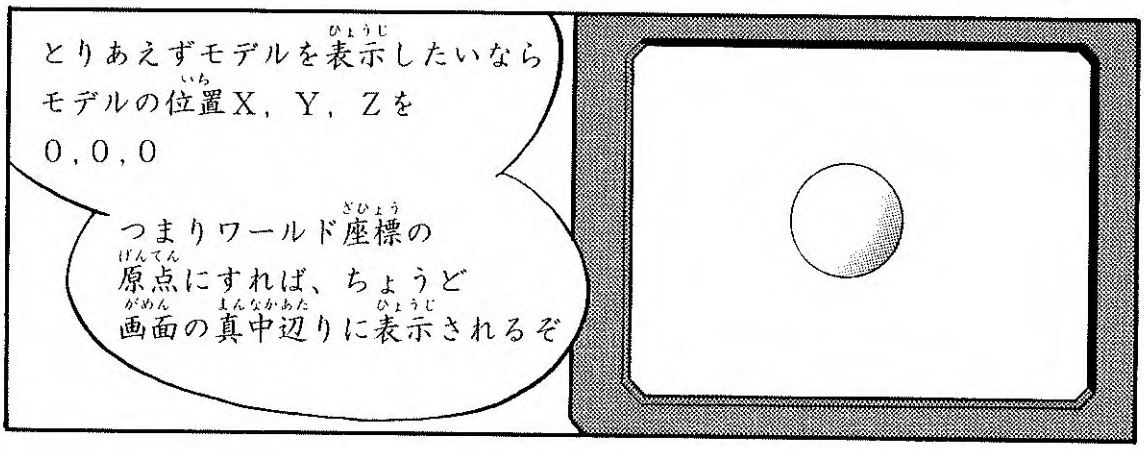

カメラといっても仮想的な
カメラだが、そのワールド
空間にあるカメラ位置
からの映像が画面上での
モデル表示ってことになる

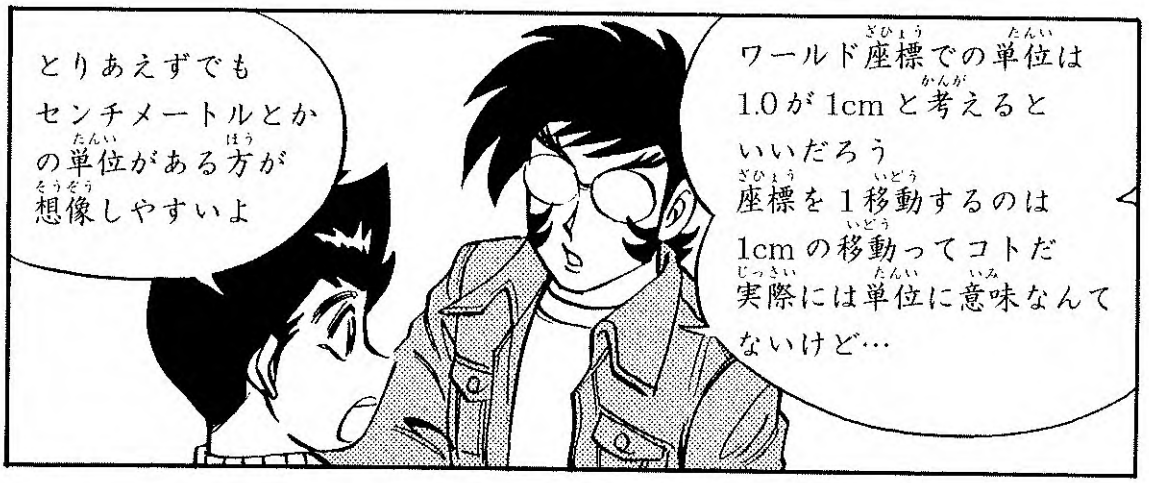

だからモデルの位置を
変更しなくてもカメラの向き
を変更してしまえばモデルは
画面外に移動してしまうぞ
ふ～ん？



MODPOS モデル番号,
ワールド座標 X, ワールド座標 Y, ワールド座標 Z

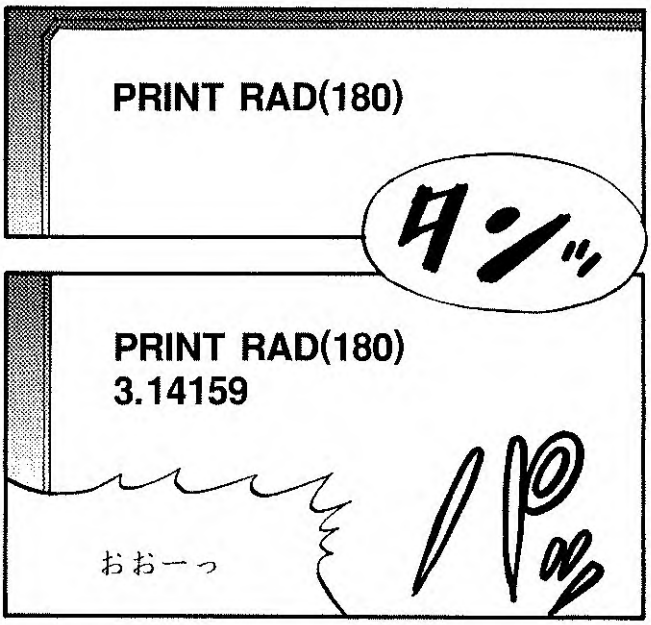

カメラが初期状態の時は
カメラはワールド座標原点から
Z軸の方に100cmくらい
離れた所からワールド座標原点
の方を向いている


とりあえずモデルを表示したいなら
モデルの位置X，Y，Zを
0，0，0
つまりワールド座標の
原点にすれば、ちょうど
画面の真中辺りに表示されるぞ


とりあえずでも
センチメートルとか
の単位がある方が
想像しやすいよ
ワールド座標での単位は
1.0が1cmと考えると
いいだろう
座標を1移動するのは
1cmの移動ってコトだ
実際には単位に意味なんて
ないけど…

MODROT モデル番号，X軸回転，Y軸回転，Z軸回転

回転を360度の単位で
扱いたいなら一つの関数を
教えようRAD関数だ


PRINT RAD(180)
タンッ
PRINT RAD(180)
3.14159
おおーっ
ハッ

引数を度で
渡せばラジアンが
返ってくる
カタ
カタ

じゃあ その関数を
使ってもらうとして
モデルを回転して
みるか
なるほど
これを使えば
いいんだな

実行・・・と

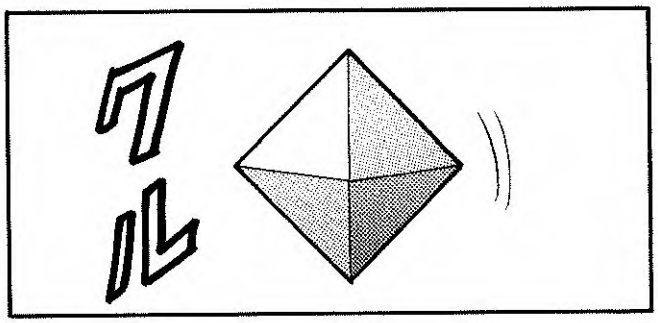
ターン

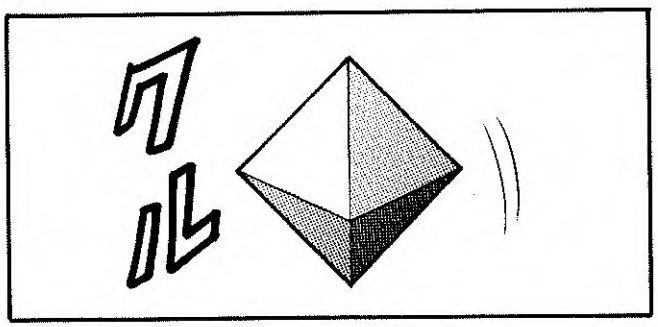
これはモデルを
回転させるプログラムだ

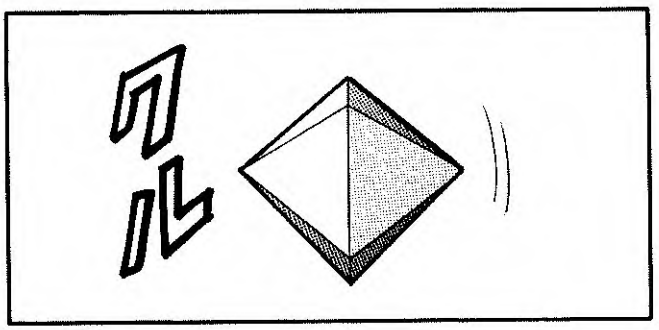
```
10 A=0
20 MODPOS 1,0,0,0
30 MODDISP 1
40 DO
50   MODROT 1,RAD(A),0,0
60   A=A+1
70   IF A>180 THEN A=A-360
80   FSYNC 1
90 LOOP
```

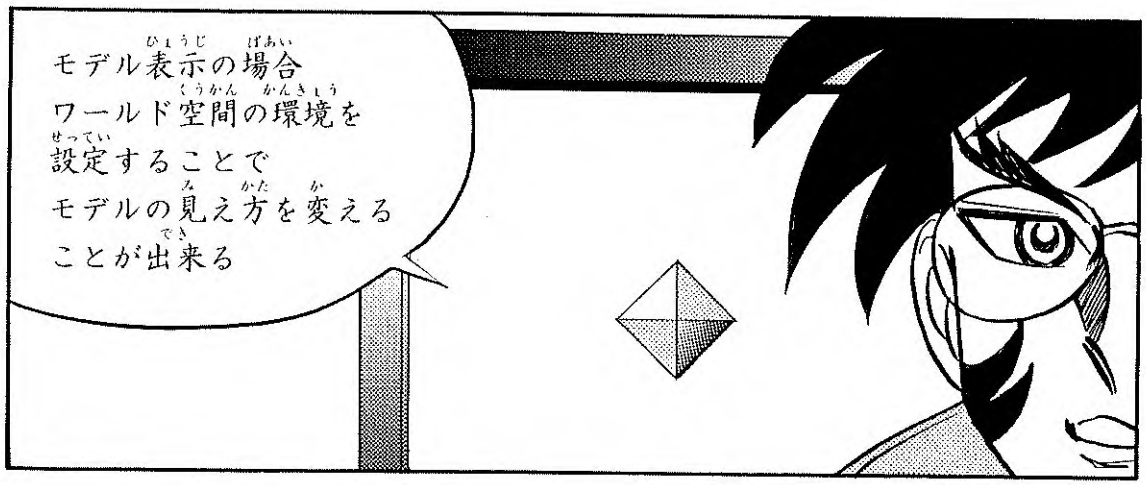
おおーっ
モデルが回転したぞ

これで
モデルの回転も
大丈夫だな



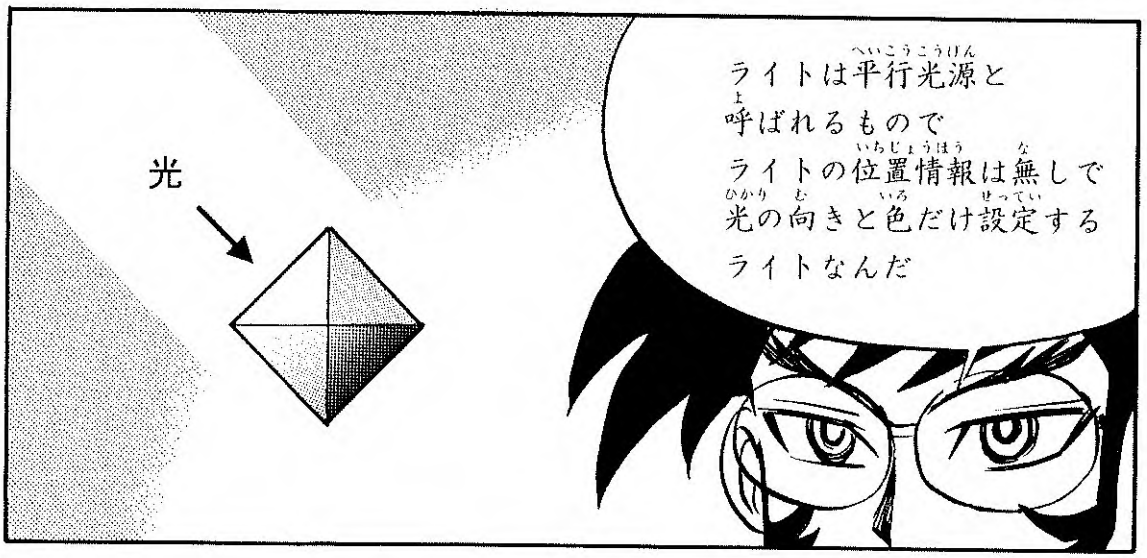




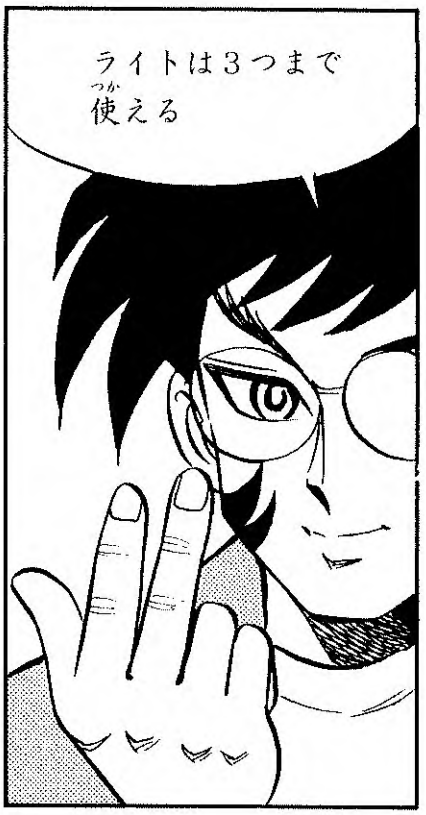

モデル表示の場合
ワールド空間の環境を
設定することで
モデルの見え方を変える
ことが出来る

ライトってのは
光を照らすライト？
そうだ

設定できる環境は
いくつかあるが
ライトを教えておこう


光
ライトは平行光源と
呼ばれるもので
ライトの位置情報は無しで
光の向きと色だけ設定する
ライトなんだ

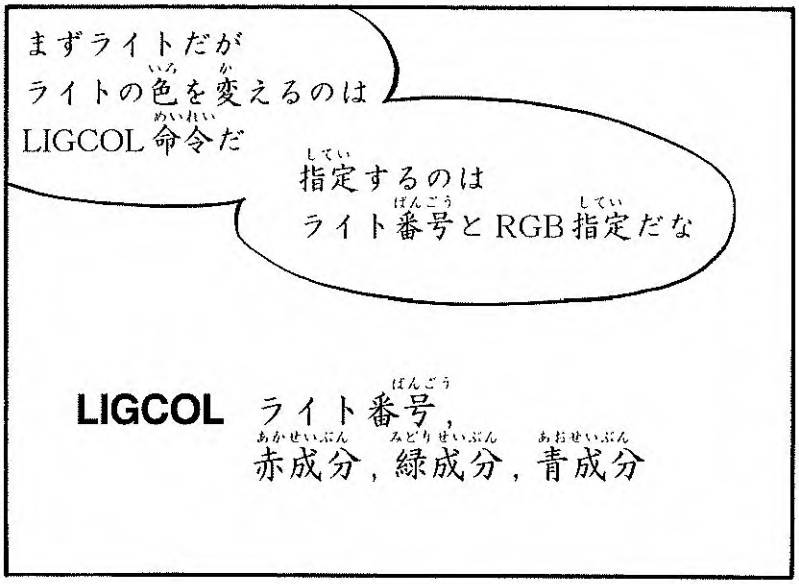
だからモデルに対して
別々の方向から
違う色のライトを
当てるような事も
出来るぞ
ふ～ん


ライトは３つまで
使える

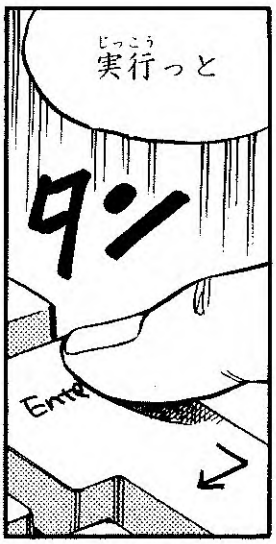
環境光の事なんだが
要するに光の当ってない部分
でも最低限どのくらい
明るくするかの設定だな

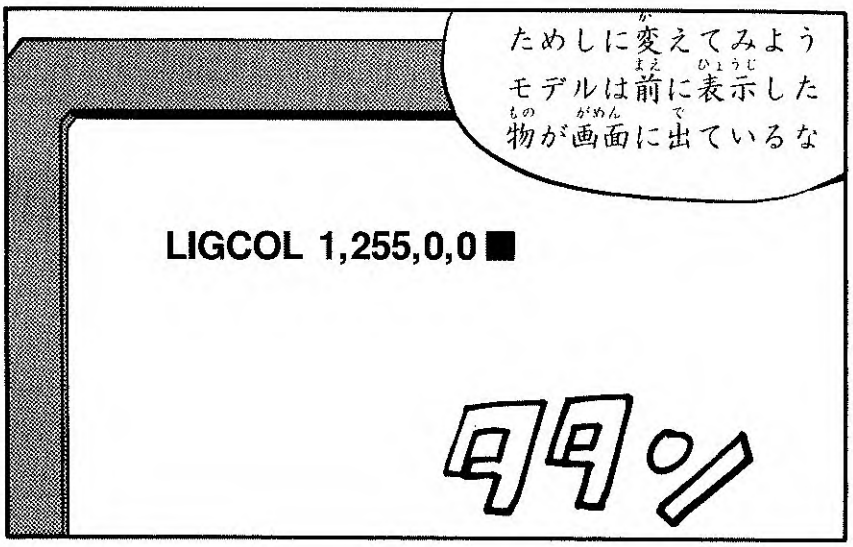
それとライトと似た
ものにアンビエント
てのもあるぞ
アンビエント？

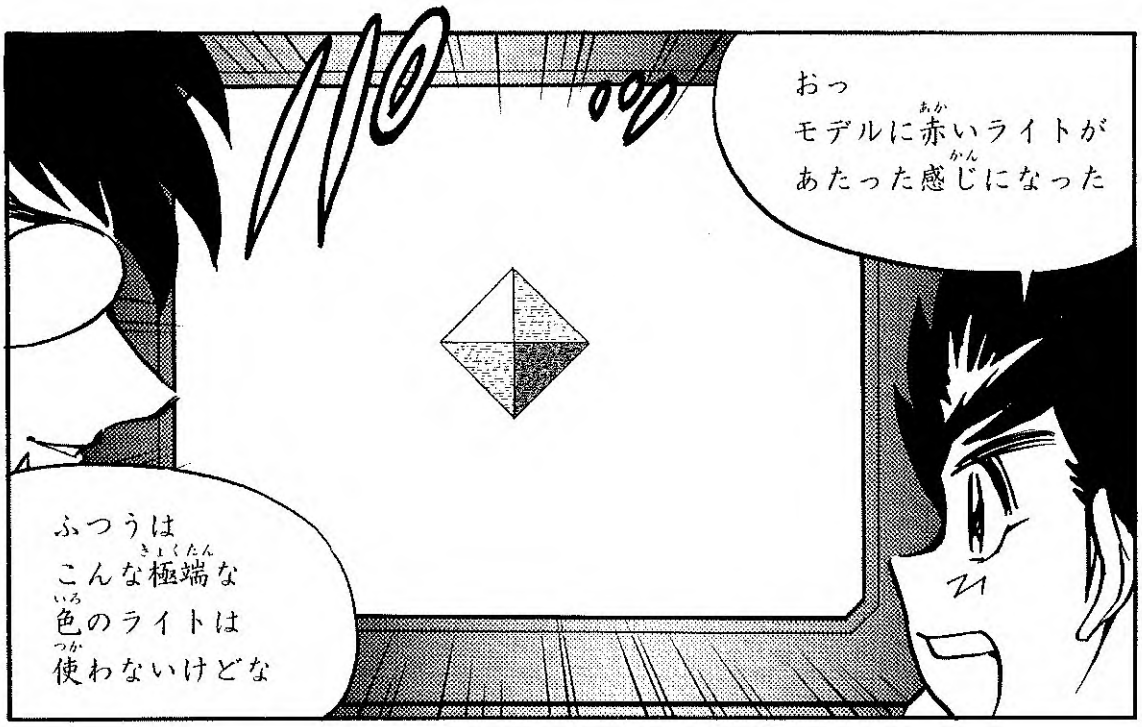
まっ黒…
アンビエントを0に
指定すると宇宙空間の
映像みたいに光の当ってない
部分がまっ黒になるんだ

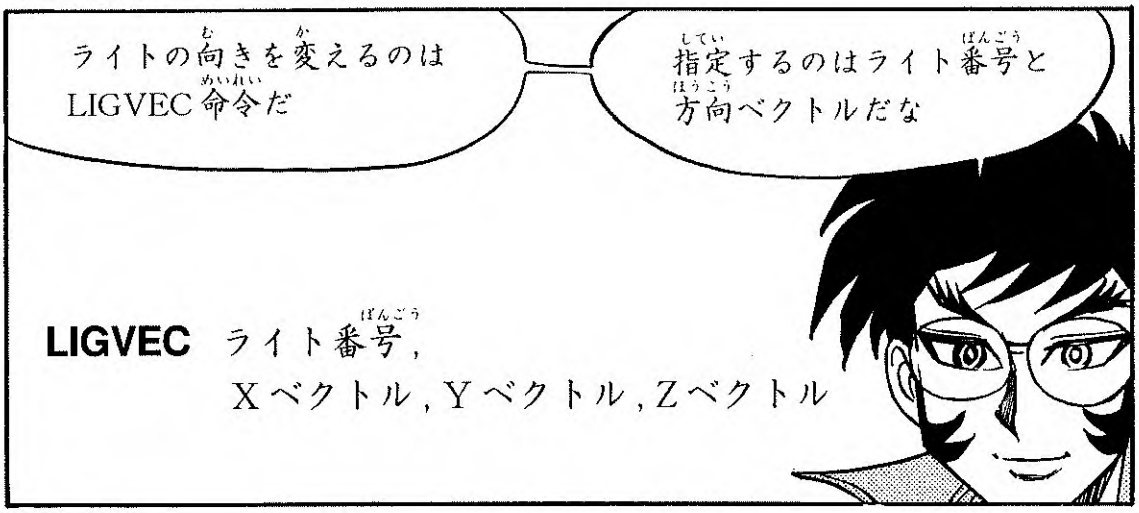

まずライトだが
ライトの色を変えるのは
LIGCOL命令だ
指定するのは
ライト番号とRGB指定だな
LIGCOL ライト番号,
赤成分,緑成分,青成分

実際に
ワールド空間の
環境を変化
させてみよう
カタッ
カタッ


実行っと
ターン
Enter

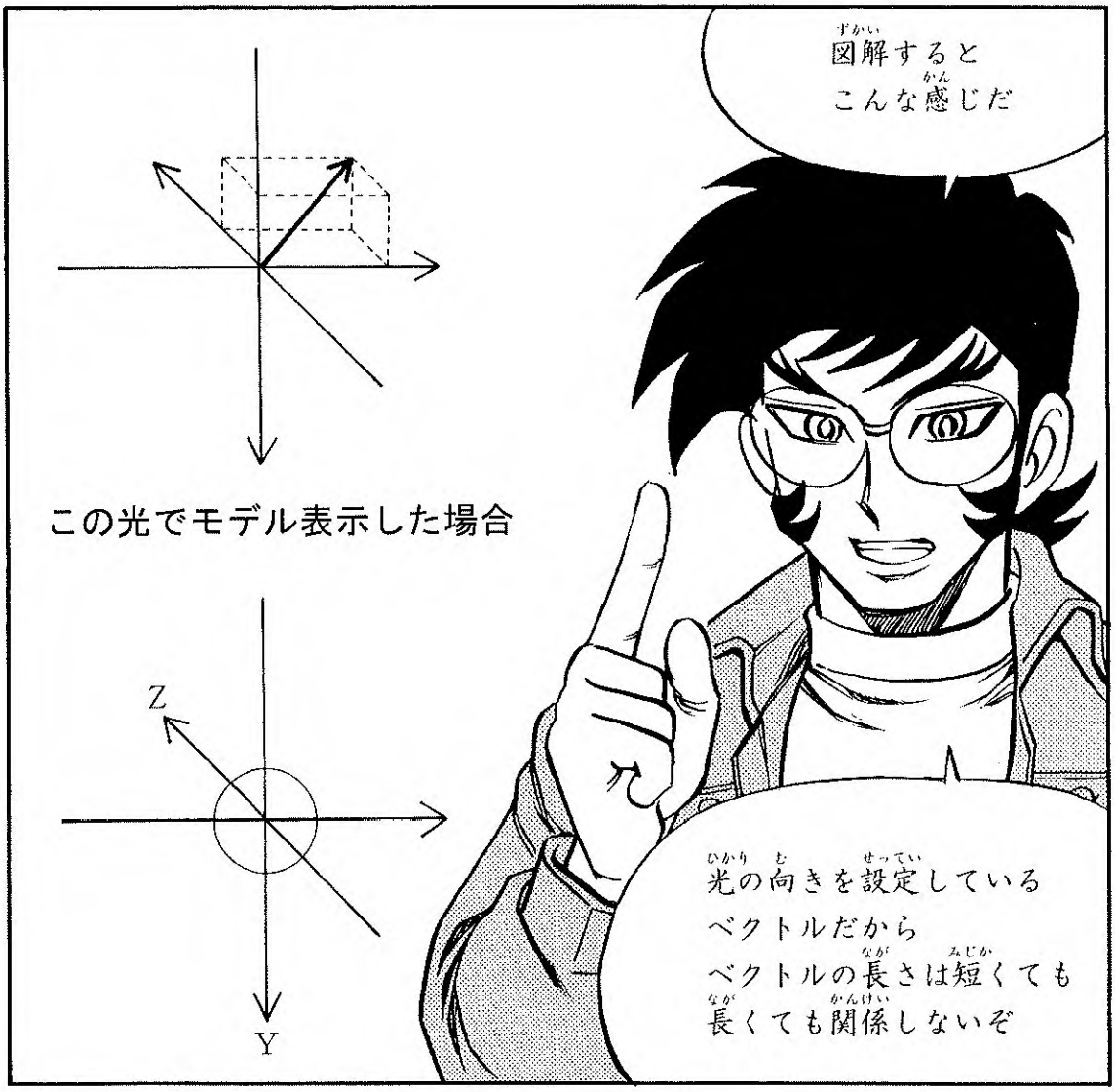

ためしに変えてみよう
モデルは前に表示した
物が画面に出ているな
LIGCOL 1,255,0,0■
ﾀﾗﾘﾗﾘｰﾝ


おっ
モデルに赤いライトが
あたった感じになった
ふつうは
こんな極端な
色のライトは
使わないけどな


ライトの向きを変えるのは
LIGVEC命令だ
指定するのはライト番号と
方向ベクトルだな
LIGVEC ライト番号,
Xベクトル,Yベクトル,Zベクトル

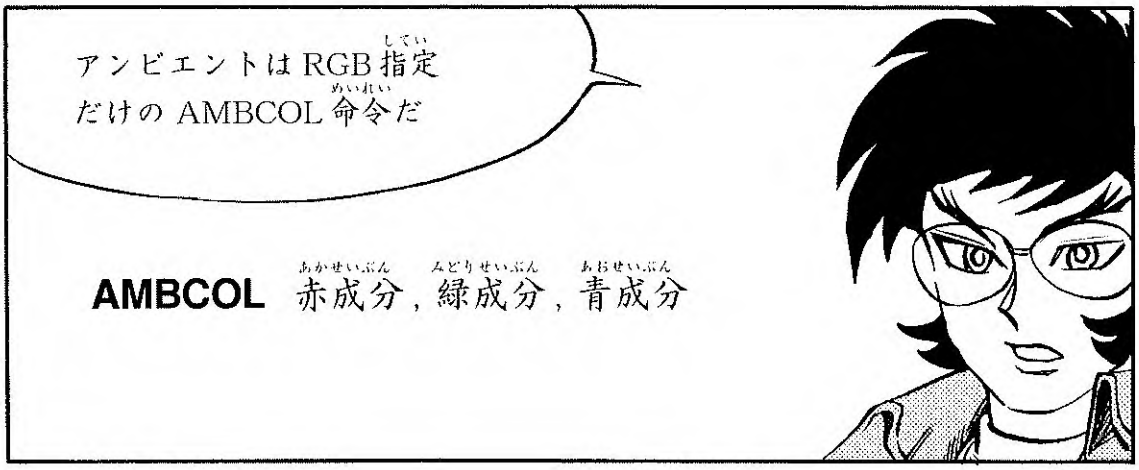
ライトのベクトルってのは
要するにX方向とY方向と
Z方向の強さの値って
とこかな

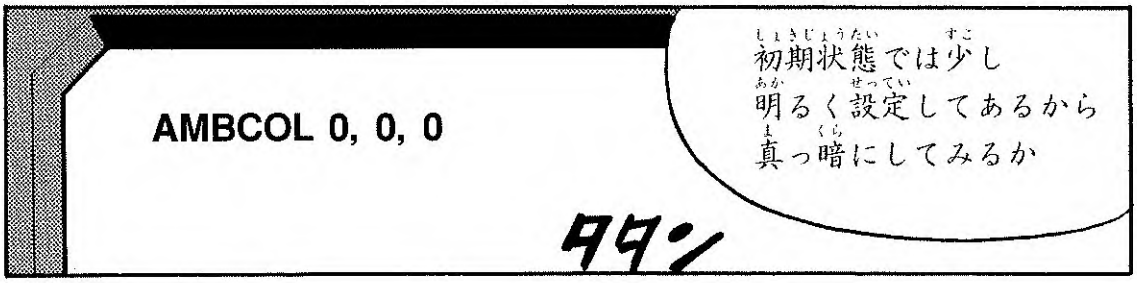
ベクトル?

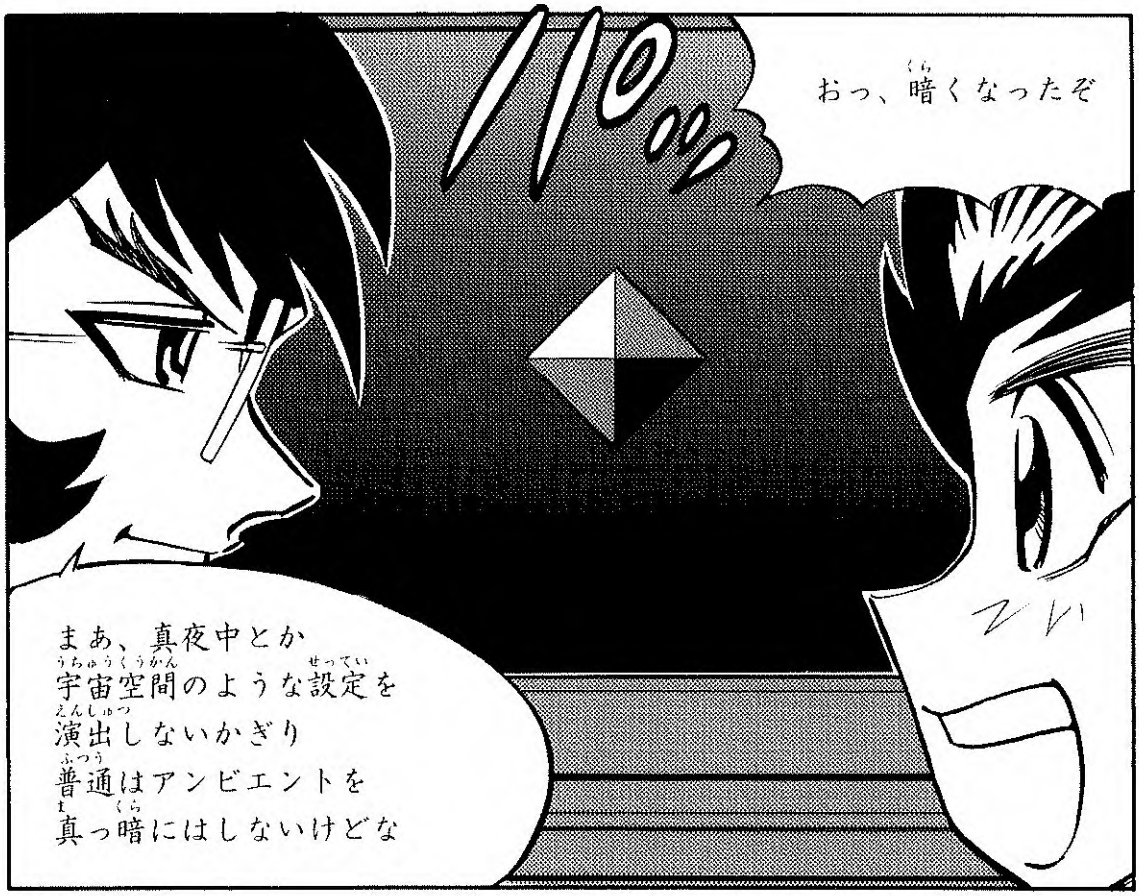

図解すると
こんな感じだ
この光でモデル表示した場合
Z
Y
光の向きを設定している
ベクトルだから
ベクトルの長さは短くても
長くても関係しないぞ

つづいてアンビエントの
設定だ
さっき言ってた
やつだね

ふんふん
了解


アンビエントはRGB指定だけのAMBCOL命令だ
AMBCOL 赤成分，緑成分，青成分


AMBCOL 0, 0, 0
初期状態では少し明るく設定してあるから真っ暗にしてみるか
99%


110%
おっ、暗くなったぞ
まあ、真夜中とか宇宙空間のような設定を演出しないかぎり普通はアンビエントを真っ暗にはしないけどな

さて
俺が教えるのは
ここまでだな
えっ！！
もう終わりなの？！

まだまだ触れていない部分も
多いが、基本は一通り教えた
他の命令はマニュアルを
見て自分で覚えていくんだ

まあ大事なことはヤル気だな
これまでに教えた事を
踏み台にして自分の力で
頑張ってみろ
うっ…
が…頑張るぜっ

あまり最初から立派な
ゲームを目指さずに
最初のうちは簡単な物から
挑戦するようにな
1つ1つのゲームを
最後まできちんと作り
上げることによって
技術と自信が付く
ものだからな

うっす！！了解！！
地道に頑張ってみるぜ

BASICSTUDIO

パワフルゲーム工房

「はじめよう! EZ-BASIC」

原作　飯塚 正樹
作画　すねや かずみ

一般の単行本のように著作権表記ではないので注意してください。

株式会社アートディンク
〒261-8701 千葉市美浜区中瀬2-6 WBGマリブウエスト22階
URL http://www.artdink.co.jp/

©2001 ARTDINK.All Rights Reserved.
"PlayStation"および"DUALSHOCK"は株式会社ソニー・コンピュータエンタテインメントの登録商標です。
ソニー製"プレイステーション 2"対応プリンター popegg 対応。
「popegg」はソニー株式会社の商標です。
その他表記されている商品名、社名は、一般に各社の商標または登録商標です。
2001年4月5日 初版発行

SLPS 20084~5